DOCUMENT RESUME

ED 267 977 SE 046 499

AUTHOR Anderson, John R.; And Others
TITLE Cognitive Modelling and Intelligent Tutoring.
INSTITUTION Carnegie-Mellon Univ., Pittsburgh, Pa.
SPONS AGENCY National Science Foundation, Washington, D.C.
PUB DATE 13 Mar 86
GRANT IST-8318629; MDA-903-85-K-0343; MDR-8470337; N00014-84-K-0064
NOTE 70p.
PUB TYPE Reports - Research/Technical (143)

EDRS PRICE MF01/PC03 Plus Postage.
DESCRIPTORS Algebra; *Artificial Intelligence; *Cognitive Psychology; *Cognitive Structures; Computer Assisted Instruction; Educational Research; Geometry; Man Machine Systems; *Models; Psychological Studies; Skill Analysis; *Tutoring
IDENTIFIERS Mathematics Education Research

ABSTRACT

Research on intelligent tutoring seeks to develop systems for automating education and to explore epistemological issues concerning the nature of the knowledge presented by tutors and how that knowledge can be learned. The ACT theory of skill acquisition and its successor (PUPS--PenUltimate Production System) provides production-system models of the acquisition of skills such as LISP programming, geometry theorem-proving, and solving of algebraic equations. The three major sections of the document discuss; (1) the cognitive theory that serves as the basis for the tutoring endeavors, (2) the model-tracing methodology and how it derives from the cognitive theory, and (3) issues that arise in implementing the methodology. Knowledge begins in declarative form and is used by analogical processes to solve specific problems. Domain-specific productions are compiled from the traces of these problem solutions. The model-tracing methodology has been developed as a means of displaying this cognitive theory in intelligent tutoring. Implementation of the methodology involves developing a student model, a pedagogical module, and an interface. Work on tutoring and on skill acquisition have proven to be symbiotic -- each has furthered the other's development. (Author/MNS)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

U.S. DEPARTMENT OF EDUCATION
NATIONAL INSTITUTE OF EDUCATION
EDUCATIONAL RESOURCES INFORMATION
CENTER (ERIC)

This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official NIE position or policy.

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

National Science Foundation

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) Technical Report No. ONR-86-1	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION John R. Anderson 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Personnel and Training Research Programs Office of Naval Research (Code 1142PT)
6c. ADDRESS (City, State, and ZIP Code) Psychology Department, Carnegie-Mellon University, Pittsburgh, PA 15213	7b. ADDRESS (City, State, and ZIP Code) 800 North Quincy Street Arlington, VA 22217-5000
8a. NAME OF FUNDING / SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-84-K-0064
8c. ADDRESS (City, State, and ZIP Code) Personnel Training and Research Programs Arlington, VA 22217	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 61153N; PROJECT NO. RR04206; TASK NO. RR04206-0A; WORK UNIT ACCESSION NO. NR 667-530

11. TITLE (Include Security Classification)
Cognitive Modelling and Intelligent Tutoring (Unclassified)

12. PERSONAL AUTHOR(S)
J. R. Anderson, C. F. Boyle, A. T. Corbett, M. W. Lewis

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Interim	FROM 3/85 TO 3/86	86/3/17	42

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP
05	09	10

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Algebra; Cognitive Science; Analogy; Computer-Assisted Instruction; Artificial Intelligence; Geometry

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The ACT* theory of skill acquisition and its PUPS successor provide production-system models of the acquisition of skills such as LISP programming, geometry theorem-proving, and solving of algebraic equations. Knowledge begins in declarative form and is used by analogical processes to solve specific problems. Domain specific productions are compiled from the traces of these problem solutions. The model-tracing methodology has been developed as a means of displaying this cognitive theory in intelligent tutoring. Implementation of the model-tracing methodology involves developing a student model, a pedagogical module, and an interface. Issues associated with the development of each of these components are discussed. Work on tutoring and work on skill acquisition have proven to symbiotic; that is, each has furthered the other's development.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Susan Chipman	22b. TELEPHONE (Include Area Code) (202)696-4318 — 22c. OFFICE SYMBOL ONR 1142PT

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

18.

Intelligent Tutoring
Interface Design
Knowledge Compilation
Learning
LISP
Model Tracing
Production System
Skill Acquisition
User Models

Introduction

Research on intelligent tutoring serves two goals. The obvious goal is to develop systems for automating education. Private human tutors are very effective (Bloom, 1984), and it would be nice to be able to deliver this effectiveness without incurring the high cost of human tutors. However, a second and equally important goal is to explore epistemological issues concerning the nature of the knowledge that is being tutored and how that knowledge can be learned. We take it as an axiom that a tutor will be effective to the extent that it embodies correct decisions on these epistemological issues.

We chose intelligent tutoring as a domain for testing out the ACT* theory of cognition (Anderson, 1983). It was a theory that made claims about the organization and acquisition of complex cognitive skills. The only way to adequately test the sufficiency of the theory was to interface it with the acquisition of realistically complex skills by large populations of students. When we read the *Intelligent Tutoring* book edited by Sleeman and Brown (1982) it became apparent that the authors in it were explicitly or implicitly performing such tests of theories of cognition and that is was an appropriate methdology for testing the ACT* theory.

The ACT* theory has been used to construct performance models of how students actually execute the skills that are to be tutored and learning models of how these skills are acquired. The performance model is used in a a paradigm we call model tracing, in which we try to follow in real time the cognitive states that the student goes through in solving a problem. Our instruction is predicated on the assumption that when we interrupt students we correctly understand their internal states. The learning model is used to infer a student's knowledge state by tracing the student's performance across problems. This knowledge tracing (as opposed to problem-state tracing) can be used to disambiguate alternative interpretations of student behavior and for selecting problems to optimize learning.

We are currently working on tutors for beginning LISP programming (Reiser, Anderson, & Farrell, 1985), for generation of proofs in high school geometry (Anderson, Boyle, & Yost,

1985), and for solving algebraic manipulation and word problems (Lewis, Milson, & Anderson, in press). These domains were selected because they involve the acquisition of well-defined skills and we can catch students at the point where they are just beginning to learn the skill. Our LISP tutor currently teaches a successful university-level course, and our geometry tutor is in the midst of its first demonstration in the high schools--an apparently successful demonstration. We believe that these tutors owe their success to the cognitive principles from which they were derived. However, it is not the case that the cognitive principles have remained unchanged in the face of these applications. In fact, we have found reasons to reject certain assumptions of the ACT* cognitive architecture and are working with a new architecture called PUPS (for PenUltimate Production System). So, even at this early stage of our endeavor, we have seen a fairly profitable flow of influence back and forth between the theory and the application.

This paper has three major sections. The first describes the cognitive theory that serves as the basis for our tutoring endeavours. The second section describes the model-tracing methodology and how it derives from our cognitive theory. The third section discusses the issues that arise in implementing the model-tracing methodology.

The Cognitive Theory

The Performance Theory

In both the PUPS theory and its ACT predecessor, a fundamental theoretical distinction was made between declarative and procedural knowledge. This distinction borrowed its label from the distinction in AI a decade ago (e.g., Winograd, 1975) but has been fundamentally transformed to be a psychological distinction. Declarative knowledge is distinguished by the fact that the human system can encode it quickly and without commitment to how it will be used. Declarative knowledge is what is deposited in human memory when someone is told something, as in instruction or reading a text. Procedural knowledge on the other hand can only be acquired through the use of the declarative knowledge, often after trial and error practice, and is further characterized by the fact that it embodies the knowledge in a highly

efficient and use-specific way. In the theory, procedural knowledge derives as a by-product of the interpretative use of declarative knowledge. In fact, we use the term knowledge compilation to refer to the learning process which creates the procedural knowledge.

Procedural Knowledge: Productions

In the ACT* theory, procedural knowledge is represented by a set of production rules that define the expert skill. Our goal in tutoring is basically to create experiences that will cause students to acquire such production rules. It would be worthwhile to examine some examples of productions that are used in our three domains of tutoring--i.e.,LISP, geometry, and algebra. Below are "Englishified" versions of a couple of the productions that are used in the LISP tutor:

IF the goal is to merge the elements of lis1 and lis2 into a list
THEN use append and set as subgoals to code lis1 and lis2

IF the goal is to code a function on a list structure
and that function must inspect every atom of the list structure
and the list structure is arbitrarily complex
THEN try car-cdr recursion and set as subgoals
1. to figure out the recursive relation for car-cdr recursion
2. to figure out the terminating cases when the argument is nil or an atom

The first is a production that recognizes the relevence of a basic LISP function and the second is one that recognizes the applicability of a recursive programming technique. These and approximately 500 more production rules model an ideal student writing basic LISP code to solve problems that would appear in an introductory LISP textbook. These productions all have this goal decomposition character of starting with some programming goal and decomposing it into subgoals until goals are reached which can be achieved with direct code. For an extensive discussion of a model of beginning LISP programming see Anderson, Farrell, and Sauers (1984).

The character of the production rules underlying the geometry tutor are somewhat different. Below are two examples from the approximately 300 in that system:

IF the goal is to prove $\triangle XYZ \cong \triangle UYW$

and XYW are colinear
and UYZ are colinear
THEN conclude $\angle XYZ \cong \angle UYW$ because of vertical angles

IF the goal is to prove $\triangle XYZ \cong \triangle UVW$
and $\overline{XY} \cong \overline{UV}$
and $\overline{YZ} \cong \overline{VW}$
THEN set a subgoal to prove $\angle XYZ \cong \angle UVW$ so SAS can be used.

The first production makes a forward inference from what is known about a problem while the second makes a backward inference from what is to be proved. A proof is completed when a line of subgoals from the to-be-proven statement makes contact with a line of forward inferences from the givens of the problem. The production rules for forward and backward inference are contextually constrained. That is, they not only make reference to the information necessary for application of the rule but also to other information about the proof which is predictive of the aptness of that inference. Thus for instance, the first rule not only makes reference to the collinearity information which is logically necessary for application of the vertical angle rule; it also makes reference to the fact that these angles are corresponding parts of to-be-proven-congruent triangles. For more discussion of the nature of the ideal student model in geometry read Anderson (1981) and Anderson, Boyle, and Yost (1985).

The production system for the algebra tutor is again somewhat different in character from the production systems for LISP or geometry. Below are three of the production rules involved in modelling the ideal student's knowledge of distribution:

IF the equation to be solved contains a subexpression of the form
"num(exp1 + exp2)"
THEN set as a subgoal to distribute num over (exp1 + exp2)

IF the goal is to distribute num1 over exp
and "num2 var" is the first term in exp
THEN write "num3 var," where num3 is the product of num1 and num2
and set a subgoal to distribute num1 over the rest of exp

IF the goal is to distribute num1 over "+ num2"
THEN write "+ num3," where num3 is the product of num1 and num2

These rules would be invoked if, for instance, there were an expression of the form ...3(5x + 2)... somewhere in the equation to be solved. The first rule recognizes the applicability of distribution; the second writes out "15x;" and the third writes out " + 6."

The algebra rules highlight the issue of grain size which is also an issue for other production systems. We could have compacted all three of these rules into a single production rule which recognized and applied distribution to the equation in one fell swoop (as, for instance, Sleeman, 1982, does). On the other hand, we could have broken each of these steps into multiple substeps. For instance, note that we do not model the process of calculating the product of num1 and num3 into a set of substeps as it might well be implemented cognitively. Our decision about the level at which to model the student was determined by pedagogical considerations. Students entering the algebra course have their multiplication skills well-learned and do not need to be tutored on these. In contrast, students do have problems with the subcomponents of distribution and so we need to separate these out for purposes of separate tutoring.

An implication is that the production rules that we use in the algebra tutor, and indeed in the other tutors, represent only upper levels of the skill. These productions set subgoals which are met by other productions and are typically accomplished in our systems by calls to LISP code. These include such things as the actual typing of answers into the computer. The assumption is that such productions, below the level that we are modelling, are well-learned.

While the production systems for the different domains do have some features in common, they also have rather different overall structures. Our learning theory would predict that the different task structures of the different domains produce different organizations of the production rules. Generating LISP code is a design activity and lends itself to a problem decomposition structure. The search character of generating geometry proofs produces an opportunistic structure in which there can be large switches of attention among parts of the proof. The linear structure of the algebra equations and the algorithmic character of

algebra equation solving produces the symbol substitution character of the algebraic rules. One of the major functions of a tutor for a particular domain should be to communicate the ideal problem-solving structure of that domain.

Declarative Knowledge: PUPS Structures

Knowledge is not originally encoded by students in such use-specific production form but rather is encoded declaratively in what we have come to call PUPS structures. PUPS structures are basically schema-like structures which are distinguished by the fact that they have certain special slots which prove critical to their interpretive application in problem-solving. These include the function slot which serves to indicate the function of the entity represented by the structure, the form slot which indicates its form, and the precondition slot which states any preconditions that must be satisfied for that form to achieve that function. To illustrate such structures let us consider how an ideal student might encode the following fragment of text from the second edition of Winston and Horn (1984), p.24:

> The value returned by car is the first element of the list given as its argument.
>
> (CAR '(FAST COMPUTERS ARE NICE))
> FAST

This Winston and Horn example is interesting because it contains a nice juxtaposition of some abstract instruction with a specific example. However, the PUPS encodings of the two (given below) are basically structurally isomorphic. The abstract encoding of **car** contains a prerequisite pointer to **structure** that indicates how **car** is to be used. The **example** structure has the same form as **structure**, except that an argument is specified. Two other PUPS structures encode that argument and the value returned by the example call.

car	ISA:	function
	FUNCTION:	(implements first)
	FORM:	(text **car**)
	PRECONDITION:	(type **structure**)
structure	ISA:	lisp-code
	FUNCTION:	(calculate (first **arg**))
	FORM:	(list **car arg**)
example	ISA:	lisp-code

	FUNCTION:	(illustrate **car**)
		(calculate (first **lis**))
	FORM:	(list **car lis**)
lis	ISA:	list
	FUNCTION:	(argument-in **example**)
		(hold (**fast computers are nice**))
	FORM:	(list '(**fast computers are nice**))
fast	ISA:	atom
	FUNCTION:	(value-of **example**)
		(first **lis**)
	FORM:	(text **fast**)

The structures above represent the outcome of successful encoding of the text; however, it should be stressed that there is a lot of room for incorrect encoding. Incorrect encoding of text into PUPS structures is what goes by the name of "misunderstanding". Clearly, a critical issue for learning is correct interpretation of the instruction. One problem with virtually all instructional material is that it omits many things that the student needs to know in order to perform the tasks, and the student is left to figure them out by trial and error experimentation. One of the payoffs in developing an ideal student model, even before it is used in tutoring, is that it provides a cognitive analysis of what the student really needs to know. Instruction can then be designed to communicate that. In our work we have found that instructional materials designed to communicate all the information in the ideal model (and not waste prose communicating non-information) are more effective than standard texts even without a tutor. This emphasis on economy and focus in instruction has been confirmed by a number of other researchers (Carroll, 1985; Reder, Charney, & Morgan, in press).

However, we believe that it is not possible to avoid all or even most misinterpretations. In communicating unfamiliar material there is the inevitable difficulty of the student being weak on the key concepts. For instance, we have never observed any student to go from reading any textbook on LISP to practicing that knowledge without errors. One important role for a tutor is to monitor for these errors of misunderstanding and correct them as they show up in the performance of a task.

Interpretive Use of Declarative Knowledge

We assume that the declarative PUPS-structures illustrated above are deposited in memory essentially as the product of language comprehension. It is important that the necessary structures get encoded correctly, but this is by no means the end state of the learning process. These structures do not directly lead to any performance and it is necessary to interpret them to get performance. This interpretive process is of high demand cognitively and is a major cause of slips in performance (Anderson & Jeffries, 1985; Norman, 1981). It is important to create productions like the ones in the ideal model which will automatically apply the knowledge.

There is essentially a double loop of inefficiency promoted by interpretive use of declarative knowledge. The inner loop involves the analogical application of the declarative knowledge encoded in PUPS-structures to a new domain to produce a step of problem solution. It is costly in terms of the amount of information that must be held in working memory to compute this analogy. So, for instance, a student might go through a prolonged effort trying to map the general statement of the side-angle-side postulate to a specific problem (Anderson, 1982). The outer loop involves a search through the problem space defined by these steps for a problem solution. So, for instance, a student might search through all the postulates for proving the angles congruent: side-side-side, side-angle-side, etc. While it is not possible to entirely avoid search, the productions in the ideal model have features built into them that greatly cut down on this search. The example productions we displayed earlier for geometry illustrate this in that they included heuristic tests that checked for likelihood that a rule of inference would contribute to a final proof.

Analogy

To illustrate the analogy process, suppose the student has the goal of getting the first element of the list (A B C). This is represented by the PUPS structures below:

goal	ISA:	lisp-code
	FUNCTION:	(calculate (first lis2))
	FORM:	?

lis2	ISA:	list
	FUNCTION:	(hold (a b c))
	FORM:	?

As is typically the case in the PUPS representation of a problem-solving situation we have PUPS structures with functions represented but forms empty. The goal is to create forms that satisfy the functional specification. Both of these forms can be calculated by analogy to the earlier PUPS structures created from comprehension of the Winston and Horn instruction. Using **example** as the source for the analogy and **goal** as the target, PUPS creates the following analogy:

function(source):form(source)::function(target):?

lis from the example is mapped to **lis2** from the target and the specification (LIST **CAR lis2**) is created for the form slot. A similar analogy between **lis** and **lis2** leads to the description (LIST '(**A B C**)) for the form slot of **lis2**. This constitutes a solution to he problem.

Knowledge Compilation

What we have just described is a solution by analogy for a specific example problem. This analogy process is costly in terms of computing the mapping. It will also only work when there is an example at hand. Knowledge compilation tries to analyze the essence of this solution and produce a production rule that can produce this solution at will. Basically, it does this by looking at the situation before and the situation after and creating a production rule that maps one onto the other. Essential to knowledge compilation is diagnosing what was critical in the before situation and what is critical in the solution. This depends on the semantics of the PUPS structure. The result of the compilation process for this example is

IF	the goal is to get the first element of =list
THEN	type (car =list)

The knowledge compilation process that produced this has to know about the

correspondences computed in calculating the analogy. Thus, this learning mechanism has built into it knowledge of how PUPS structures are interpreted in analogy.

Search

A second thing knowledge compilation will do is eliminate some of the relatively blind search that characterizes early problem solving. Consider the diagram in Figure 1, which shows a problem that appears early in the geometry problem sequence. The student is given that two sides of the triangles are congruent and must try to prove that the triangles are congruent. At this point the student has only learned of the side-side-side and side-angle-side postulates for proving triangles congruent. One student, not atypical, was observed to (1) try side-angle-side but fail because there is not an angle congruence; (2) try side-side-side but fail because only two sides are given as congruent; (3) apply the definition of congruence that the measure of $\overline{AD}$ is equal to the measure of $\overline{CD}$; (4) Apply the reflexive rule to infer $\overline{AD}$ is congruent to itself; and, (5) finally, applying the reflexive rule, to infer that $\overline{BD}$ is congruent to itself. This last step was the key one that allowed the student to immediately apply the SSS rule to achieve his goal. It seemed that the subject engaged in a random search of legal operators until he came across one that was useful.

Insert Figure 1 about here

Knowlege compilation creates rules that skip over the steps that were not relevant to the final solution and try to produce a rule that connects key features in the original situation with the ultimately useful operator. The rule that should be produced in this case is:

IF the goal is to infer $\triangle XYZ \cong \triangle UYZ$
THEN infer $\overline{YZ} \cong \overline{YZ}$ because of the reflexive property of congruence

Note this rule is not specific to the solution of this problem by SSS nor to the fact that there are already two sides proven congruent. This is what we noted of our subject: He emerged from this episode with a tendency to infer the shared side of two triangles is congruent to itself whenever he set as his goal to prove these triangles congruent.

This geometry example illustrates the general features of learning from search: If the student applies a number of operators and some of the operators prove successful--in the geometry example a number of inferences have been applied and one has proven part of the final proof --the student can encode in declarative structures how the operators achieved their successful function and this information can be compiled into new production rules. It is critical that the students properly encode their experience and this is again where tutors can be critical--by assuring the proper encoding of the experience. So for instance, in the reflexive case discussed above, if the student represented the function of the rule as establishing side-side-side he would have created too specific a rule. On the other hand, if he represented it as just making a legal inference he would have created too general a rule.

Strengthening

In addition to knowledge compilation, there is a simple strengthening of declarative and procedural knowledge with use. As knowledge becomes strengthened it comes to be applied more rapidly and reliably. There is ample empirical evidence for such a simple learning process in humans although its exact nature is in some dispute (Anderson, 1982). The major implication of a strengthening-like process for tutoring concerns the introduction of new knowledge. As the execution of acquired knowledge becomes more proficient there is more capacity left over to properly process the new knowledge.

Other Learning Mechanisms?

An important characteristic of this model is what it does not contain. Unlike the ACT line of learning theories there are no inductive learning mechanisms that automatically compare the current situation with past situations and try to form generalizations and discriminations about when rules will and will not apply. This is not to say that subjects do not engage sometimes in inductive behavior as a conscious problem-solving activity--they certainly do. Rather the claim is that there is not an automatic learning mechanism of the status of compilation and strengthening. Generalizations and discriminations are declarative

knowledge structures produced by problem solving productions rather than productions produced by automatic learning mechanisms. There is a fair amount of evidence that people are aware of their inductive generalizations and discriminations (Lewis & Anderson, 1985; Dulany, Carlson, & Dewey, 1984).

This has major implications for instruction. Rather than leaving students to induce generalizations and discriminations from carefully juxtaposed examples, which would have been the pedagogical implication of ACT, one should simply tell the student what the critical features are. Thus, if a student is overusing the vertical angle inference he should be told the circumstance under which he wants to use it. This is not to argue that examples are not important, but they should be annotated with information about what they are supposed to illustrate.

Converting Theory to Tutoring: Model Tracing

This theory of knowledge acquisition is radical in the juxtaposition of its simplicity and its claim to completeness. To review, learning in the theory involves:

1. Acquisition of new declarative knowledge by the processing of experience through existing productions (eg. for language comprehension).

2. Application of declarative knowledge to new situations (i.e., situations for which productions do not exist) by means of analogy and pure search.

3. Compilation of domain-specific productions.

4. Strengthening of declarative and procedural knowledge.

Probably there is little controversy that these things (or something very similar to them) are involved in knowledge acquisition, but the issue is whether these assumptions are sufficient to account for all knowledge acquisition. The question is how do we put that

theory to test. As argued in detail elsewhere (Anderson, in preparation) the tutoring work is a methodology for testing the theory. Since the design of the tutors is based on the theoretical analysis, the success of the tutors is one test of the theory. Moreover, one can ask whether the course of learning displayed with the tutor is in detail as predicted by the theory.

The simplicity of the underlying theory maps onto a rather straight forward tutoring methodology that we call model tracing. The basic idea is to use the learning model to trace the student's knowledge state across problems and to use the performance model to trace the student's problem state within a problem. Problems and accompanying instruction are selected to practice the student on productions that are diagnosed as weak or missing in the student's knowledge state. Instruction is generated that the student should be able to map onto the solution of a problem to enable the student to correctly interpret that solution. Given this structuring of the learning situation, we trust the automatic learning mechanisms in (1)-(4) above to move the student forward on an optimal learning trajectory.

First, we will give some examples of this model-tracing methodology. Then, we will discuss some issues in implementing it.

The LISP Tutor

The LISP tutor is based on our earlier efforts to model learning to program in LISP (Anderson, Farrell, & Sauers, 1984). Table 1 contains a dialog with a student coding a recursive function to calculate factorial. This does not present the tutor as it really appears. Instead, it shows a "teletype" version of the tutor where the interaction is linearized. In the actual tutor the interaction involves updates to various windows. In the teletype version the tutor's output is given in normal type while the student's input is shown in bold characters. These listings present "snapshots" of the interaction; each time the student produces a response, we have listed his input along with the tutor's response (numbered for convenience). The total code as it appears on the screen is shown, although the student has added only what is different from the previous code (shown in boldface type). For

instance. in line 2 he has added "zero" as an extension of "(defun fact (n) (cond ((."

Insert Table 1 about here

In the first line, when the subject typed "(defun", the template

(defun <name> <parameters> <body>)

appeared. The terms in <-> angle brackets denote pieces of code the student will supply. The subject then filled in the <name> slot and the <parameters> slot and had started to fill in the <body> slot. Note that at all points, parentheses are balanced and syntax is checked. The motivation here is to remove from the student some of the cognitive load required for checking low-level syntax and to enable the student to focus on the conceptual levels.

Although the student has some difficulty with the syntax of the conditional tests in lines 1 and 2, he basically codes the terminating case for the factorial function correctly. Typically, we find students have little difficulty with terminating cases but have great difficulty with recursive cases. The dialogue after line 3 illustrates how the tutor guides the student through a design of the recursive function. Basically, it leads the student to construct a couple of examples of the relationship between fact (n) and fact (n-1) and then gets the student to identify the general relationship. Figure 2 shows the screen image at a critical point in the design of this function.

Insert Figure 2 about here

The dialogue after this point shows two errors that students make in defining recursive functions. The first, in line 4, is to call the function directly without combining the recursive call with other elements. The second, in line 6, is to call the function recursively with the same argument rather than a simpler one.

After the student finishes coding the function he goes to the LISP window and

experiments. He is required to trace the function, and the recursive calls embed and then unravel. Figure 3 shows the screen image at this point with the code on top and the trace below it.

Insert Figure 3 about here

This example illustrates a number of features of our tutoring methodology.

1. The tutor constantly monitors the student's problem-solving and provides direction whenever the student wanders off one of the correct solution paths.

2. The tutor tries to provide help with both the overt parts of the problem solution and the planning. However, to address the planning a mechanism had to be introduced in the interface (in this case menus) to allow the student to communicate the steps of planning.

3. The interface tries to eliminate aspects like syntax checking, which are irrelevant to the problem-solving skill being tutored.

4. The interface is highly reactive in that it makes some response to every symbol the student enters.

It is interesting to note the contrast between the LISP tutor and the PROUST system of Johnson and Soloway (1984). That system provided feedback only on residual errors in the program and does not try to guide the student in the actual coding. One technical consequence is the PROUST system has to deal with disentangling multiple bugs. Since the LISP tutor only corrects errors immediately, the code never contains more than one bug at a time.

The Geometry Tutor

The geometry tutor is similarly based on our earlier work studying geometry problem-solving (Anderson, 1981, 1982, 1983a). Figure 4 illustrates how the problem is initially presented to a student. At the top of the figure is the statement the student is trying to prove. At the bottom are the givens of the problem. In the upper left corner is the diagram. The system prompts the student to select a set of statements with a mouse. Then the system prompts the student to enter a rule of geometric inference that takes these statements as premises. When the student has done so, the system prompts the student to type in the conclusion that follows from the rule. The screen is updated with each step to indicate where the student is. The sequence of premise, rule of inference, and conclusion completes a single step of inference. Figure 5 illustrates the screen at the point where the student has decided to apply definition of bisector to the premise $\overline{JK}$ bisects $\angle XJY$ but has not yet entered the conclusion. A menu has been brought up at the left of the screen to enable the entry of the conclusion. It contains the relations and symbols of geometry. By pointing to symbols in the menu and to symbols in the diagram, the student can form the new statement $\angle XJK \cong \angle JKY$. We find it useful to have the student actually point to the diagram to make sure the student knows the reference of the abstract statements.

Insert Figures 4 and 5 about here

Figure 6 shows the geometry diagram at a still later point. The student has completed the bisector inference and added a plausible transitivity inference but one that proves not to be part of the final proof. At this point the student begins to flail and has tried a series of illegal applications of rules, the most recent being application of angle-side-angle (ASA) to the premises $\angle EJX \cong \angle EJY$ and $\angle EXJ \cong \angle EXK$. The tutor points out that ASA requires three premises, and so it clearly is inappropriate. Since the student is having so much difficulty, the tutor points the student to the key step in solving this problem: To prove $\triangle EJY \cong \triangle EKX$ one will have to prove $\triangle EJY \cong \triangle EJX$ and $\triangle EJX \cong \triangle EKY$ and then apply transitivity. To explain this step the tutor is going to introduce the student to a step of

backward inference. The tutor has boxed the conclusion and will step the student through how transitivity of the two triangle congruences will enable the conclusion to be proven. The student then will have the task of proving the two triangle congruences.

Insert Figure 6 about here

Figure 7 shows the state of the diagram at a still later point where the student has proven one of the triangle congruences while the other remains to be proven. It nicely illustrates how students can mix inferencing forward from the givens and reasoning backwards from the conclusions.

Insert Figure 7 about here

Figure 8 shows the completed proof in which there is a graph structure connecting the givens to the to-be-proven statement. Students find such representations of proof solutions enlightening in two ways. First, it enables them to appreciate how inferences combine to yield a proof, something they tend not to get from the traditional two column formalism. Second, the search inherent in proof generation is explicity represented. So, for instance, students can immediately identify inferences, such as the angle transitivity inference, which are off the main path.

Insert Figure 8 about here

The Algebra Tutor

The algebra tutor (Lewis, Milson, & Anderson, in press) is a more recent endeavor of ours and does not have the prior history of domain study. It reflects an attempt to see how well the methodology that we have developed transfers to a new domain. Figure 9 shows the initial interface that we have developed for the algebra tutor.

Insert Figure 9 about here

We have turned the computer's high resolution display into a sort of notebook/scratch-pad/blackboard which is used to echo tutoring interactions and store a record of the user's intermediate work. This design allows the mouse to be used as the primary input device by sensitizing regions on the screen to button activity on the mouse.

At the current time we have the following windows:

- The Tutoring Window (made to resemble a blackboard): where tutorial interactions are printed.
- The Scratch-Pad: a key-pads for generating algebraic expressions which are needed for responses to the tutoring interactions.
- The Current Equation Window: Always displays the current state of the equation transformation process.
- The History Window (made to resemble a notebook): a trace of the problem solving is recorded for on-line review and later printing by the student.

Our goal was to make the interactions with the interface as easy as pencil and paper. When generating responses to the tutorial dialog, students need not type anything on the keyboard. In fact, in most cases, the response can generated by pointing to parts of the existing expressions in the current equation window. By using pointing instead of requiring students to regenerate complete expressions when only part might be changed, we can eliminate a good number of slips in the early stages of learning algebra. Accidently dropping a negative sign or forgetting to bring down a term are common errors when moving from one equation to the newly transformed equation. By having students point to expressions and having the system bring down the un-transformed parts of the equation intact, we can minimize the chance that these low level errors would interfere with the goal of the lesson, e.g., practicing the distributive law in equation solving.

Summarizing the Model-Tracing Methodology

The most distinctive feature of the model-tracing methodology is how closely it sticks to the target task it is supposed to be tutoring. Declarative instruction takes the form of commented examples of correct problem-solving and comments on the student's problem

solution. This is the kind of instruction that we believe can be effectively turned into procedures. The major activity of the tutor is monitoring students' problem solving. We attempt to create highly interactive interfaces that quickly let the students know when their solutions deviate from ideal solution behavior and just where they deviate. This kind of instructional environment has a highly procedural flavor and contrasts with the more abstract and declarative instruction in some tutoring efforts (e.g., Collins, Warnock, and Passafuime, 1975). This reflects fundamental differences about the nature of the knowledge to be communicated and about how that knowledge is communicated.

Implementing the Model-Tracing Methodology

A major prerequisite to implementing a model-tracing tutor is to create all the production rules that will be involved in the tracing. A significant subtask here is adding an adequate set of buggy rules to the student model in order to be able to account for the errors we see. In our experience the best we have been able to do is to account for about 80% of the errors--the remaining being just too infrequent and too removed from the correct answer to yield to any analysis. One approach to coding the systematic errors has been simply to observe the errors students make with our tutor, try to understand their origin, and code the inferred buggy productions one by one into the system. In more recent work such as in our algebra tutor we are trying to generate these errors on a principled basis something like in the notable work on subtraction (Brown & Burton, 1978; Brown & VanLehn, 1980) and on algebra (Matz, 1982).

Given a production set which can model the range of behaviors we see in our students, our tutor design then can be decomposed into three largely independent modules. There is the student module which can trace the student's behavior through its non-deterministic set of production rules. There is the pedagogical module which embodies the rules for interacting with the student, for problem selection, and for updating the student model. The separation between student model and pedagogical model is similar to the separation of instruction from the instruction in a number of tutoring systems, including Brown, Burton, and

DeKleer (1982) and Clancey (1982). Finally, there is the interface which has the responsibility for interacting with the student. As a software engineering issue, these three components can be developed separately with the pedagogical module responsible for interaction-- getting interpretations and predictions from the student module and making requests to the interface for interaction. While each module is complex, dividing a major software project into three independent components is a major step in the direction of tractability. Much of the subsequent discussion will be organized around issues involving each of the components.

The Student Module

The basic responsibility of the student model is to deliver to the tutor an interpretation of a piece of behavior in terms of the various sequences of production rules that might have produced that piece of behavior. The obvious methodology for doing that is to run our non-deterministic student model forward and see what paths produce matching behavior. While there are complexities and efficiencies that have been added to this basic insight this is the core idea. The rest of the discussion of the student model is concerned with issues raised in trying to implement this core idea.

Nondeterminacy in the production sequence is a major source of problems in implementing the model-tracing methodology. We face nondeterminacy whenever multiple productions in the student module produce the same output. (For instance, in the algebra tutor the student says he wants to apply distribution, and there are multiple possible distributions in the equation.) A special case of this is when productions produce no overt output as when a student is doing some mental calculating or planning. What to do in the case of such planning nondeterminacy is an interesting question. The set of potential paths can explode exponentially as the simulation goes through unseen steps of cognition. Also, the potential for actually effectively tutoring these steps is weakened the greater the distance between the mental mistake and the feedback on that decision. Therefore, one is naturally tempted to query the student as to what he is thinking–that is, to force an association of

some output with the mental steps. On the other hand, it is difficult to design an interface which can trace planning in a way that does not put an undue burden on the student. Students often resent even giving vocalized answers to the question "what are you thinking about" and there is reason to believe such simultaneous report generation may interfere with the problem-solving (Ericsson and Simon, 1984). The interaction in the LISP tutor trace in which the tutor tries to work through the recursive plan for factorial is one instance of our effort at tracing planning. While we have some evidence that such interactions help, students report that they do not like being slowed down by having to go through menu interactions. In the algebra tutor students are encouraged to use the screen interface to work out mental calculations. This seems to be working out fairly well.

Another example of the problem created by non-determinancy is that misunderstandings and slips can often produce the identical behavior. For instance, students can confuse CONS and LIST in programming either because they really do not understand the difference or as a result of a momentary lapse (Anderson & Jeffries, 1985). The student model must be capable of delivering both interpretations to the tutor, leaving to the tutor the task of assessing the relative probability of the two interpretations and deciding what remedial action should be taken.

A major complication we face when we try to trace a student's problem-solving is that running a production-system in real time can create serious problems. Students will not sit still as a system muddles for minutes trying to figure out what the student is doing. They will not pace their problem-solving to assist the diagnosis program. Interestingly, our observation has been that human tutors have problems with real-time diagnosis and one of the dimensions on which human tutors become better with experience is real-time diagnosis.

Production systems, for all their advantages, are by and large not the most efficient way to solve problems. Analysis has typically shown that their computation time tends to be spent in pattern matching. The inherent computational problems of production systems are exacerbated in tutoring for a number of reasons:

(1) The grain size of modeling is often smaller than would be necessary in expert-system applications, and the complexity of the production patterns required to expose the source of student confusions is often considerable.

(2) The system has to consider enough productions at any point to be able to recognize all next steps that a student might produce. This contrasts with many applications where it is sufficient to find a production that will generate a single next step.

(3) Often it is not clear which of a number of solution paths a student is on and the production system has to become non-deterministic to enable a number of paths to be traced until disambiguating information is encountered.

The production systems we have produced have all involved variations on the RETE algorithm developed by Forgy (1982) for pattern-matching which has supported many of the OPS line of production systems. However, we have not had good success with simply using OPS as our expert system because the pattern-matching for each domain has special constraints upon which we have had to try to optimize. Anderson, Boyle, and Yost (1985) contains a discussion of this issue for the domain of geometry.

A major issue in designing the pattern matcher for a domain is to decide how much detail of the actual problem should be represented. For instance, if one was developing an algebra tutor it is useful to have different representations for to the following two expressions during the early stages of teaching factoring:

2AB + 4A
2BA + 4A

There is evidence that the first expression can be more easily factored into 2A(B + 2) than can the second expression: Commutativity of multiplication is not automatic in many students, and the common factor of 2A might not be seen in the second expression above. On the other hand, when we look at students who have mastered algebra and are learning calculus, it is no longer necessary to represent the distinction between these two forms. This means that in calculus we can use certain "canonicalizations" that simplify the pattern

matching and reduce the number of productions.

The computational cost associated with implementing such production systems has a space as well as a time dimension. The number of productions can be on the order of thousands to tutor a domain and the RETE algorithm can be space expensive storing partial products of pattern matching.

Of course, it is an open question just how efficient in time and space we can make our production system implementations. In their current form they are just within the threshold of acceptability, which is to say students are barely satisfied with the performance of a machine like a DandyTiger with over three meg of memory. However, there are reasons for us not to be satisfied with this performance. In the first place such machines are still a good deal beyond the range of economic feasibility. Secondly, efficiency issues impact on the range of topics we handle. This manifests itself in a number of ways:

(1) Problems tend to become more costly as they become larger even if the larger problems involve the same underlying knowledge. Therefore, there is a artificial size limit on the problems we tutor students on.

(2) Progress into more advanced topics is as much limited by dealing with the added computational burden posed by these topics as with adequately understanding and modelling the domain.

(3) The actual tutoring interactions become limited by the need to reduce non-determinacy. For instance, some of our tutors force a particular interpretation of the student's behavior on the student, rather than waiting until the student generates enough of the solution to eliminate the ambiguity.

Compiling the Model-Tracing

If one looks at all possible sequences of productions that can be generated in any of our models, one finds that it defines a problem-space of finite cardinality. That cardinality

can be quite large but often simply because we are looking at different permutations of independent or nearly independent steps in a problem solution. This suggests that if we are clever in our representation of the problem space we need not dynamically simulate the student in order to interpret him. Rather, we can generate beforehand the problem-space and just use the student's behavior during problem solving to trace through this pre-completed problem space. Given the cost of real-time simulation with a production system, this seems that it might be a worthwhile step. (In fact, we obtained a 50% performance improvement in our LISP tutor by a partial implementation of this step.)

There are other advantages to having the complete problem space compiled in advance of the actual tutoring session. This makes it easy for the tutor to look ahead and see where a step in the problem solution will lead. Often a production rule will be favored by the ideal model but in fact not lead to a solution. For instance, there are geometry problems where even experts make certain inferences which do not end up as part of the final proof. It is the sort of heuristic inference which 9 times out of 10 is good but not in this case. If the tutor recommended dead-end steps just because the ideal model makes them, the student would quickly loose faith in the tutor. Human tutors also tend to look ahead to make sure that their recommendations lead somewhere.

The Pedagogical Module

One interesting observation about this framework is that it is possible to decouple the pedagogical strategy from the domain knowledge. Domain knowledge resides in both the student model and the interface. It is the pedagogical module that relates the two together and which controls the interaction. This module does not really require any domain expertise built into it. It is concerned with (1) what productions can apply in the student model, not the internal semantics of the productions; (2) what responses the student generates and whether these responses match what the productions would generate, not what these responses mean; and, (3) what tutorial dialogue templates are attached to the productions, not what these dialogues mean.

We are in fact working on a new PUPS-based tutor which is trying to implement this realization in the limited domain of tutors for three programming languages--LISP, ADA, and Prolog. We have built student models for different programming domains independent of tutoring strategy and have built different tutors to implement variations on tutoring strategy independent of domain. Specific tutors can be generated by crossing the tutorial module with the domain module without tuning one to another.

There are theoretical reasons for believing that we can create domain-free tutoring strategies and that the optimal tutoring strategy will be domain free. Basically, our theory of human skill acquisition leads us to believe that the basic learning principles are domain free. The optimal tutoring strategy would simply optimize the functioning of these learning principles.

However, in our current running systems we have built a separate tutor for each domain. While it is not the case that the tutoring strategies they implement are identical, they are quite similar and we have claimed publically that they are attempts to embody a strategy based on the ACT learning theory (Anderson, Boyle, Farrell, & Reiser, in press). However, in retrospect it is becoming clear that some of the features of these strategies were determined by issues of technical feasibility in a way that they need not have been. It is useful to identify what the features of the common tutoring strategy are and what the variations on the strategy could be. It will become clear that, when we look at any dimension of tutoring, there are conflicting considerations as to what the optimal choice should be.

(1) Immediacy of Feedback.

The policy on immediacy of feedback is well-illustrated by the LISP tutor. The LISP tutor insists that the student stay on a correct path and immediately flags errors. This minimizes problems of indeterminacy. There are a number of reasons for desiring immediacy of feedback besides this technical one. First, there is ample psychological

evidence that feedback on an error is effective to the degree that is given in close proximity to the error. The basic reason for this is that it is easier for the student to analyze his mental state that led to the error and make appropriate correction. Second, immediate feedback makes learning more efficient because it avoids long episodes in which the student stumbles through incorrect solutions. Third, it tends to avoid the extreme frustration that builds up as the student struggles unsuccessfully in an error state.

However, we have discovered a number of problems with the use of immediate feedback:

(a) The feedback has to be carefully designed to force the student to think. If at all possible, the feedback should be such that the student is forced to calculate the correct answer rather than just being given the answer (Anderson, Kulhavy, & Andre, 1972). It is important to learning that the student go through the thought processes that generate the answer rather than copy the answer from the feedback.

(b) Sometimes students would have noticed the error and corrected it if we just gave them a little more time. Self-correction is preferable when it would happen spontaneously.

(c) Students can find immediate correction annoying. This is particularly true of more experienced students. Thus, novice programmers generally liked the immediate feedback feature of our LISP tutor whereas experienced programmers did not. While our goal is not to produce positive affective response, it probably does have some impact on learning outcome.

(d) Often it is difficult to explain why a student's choice is wrong at the point at which the error is first manifested because there is not enough context. To consider a simple example, compare a student who is going to generate (append (list x) y) where (cons x y) is better. It is much easier to explain the choice after the complete code has been generated rather than after (append.... has been typed.

There is no reason why the model-tracing paradigm commits us to immediate feedback,

although as noted there are psychological reasons for choosing it. One of the variations we would like to explore with our PUPS-tutor is a system that gives feedback after "complete" expressions like (append (list x) y). This will enable the student some opportunity for self correction if the correction occurs before the expression is complete and provide a larger context for instruction. On the other hand the distance between error and feedback will still be limited. We have also thought about varying the amount of code we would take in before instruction as a function of experience.

(2) Sensitivity to Student History.

By and large we have used what we have called a "generic" student model in our tutoring. At each point in time we are prepared to process all the production rules that we have seen any student use, correct or buggy. If students make an error we give the same feedback independent of their history. The only place we show sensitivity to student history is in presenting remedial problems to students who are having difficulties. It is relatively easy to implement a generic student model, and the question is whether there is any reason to go through the complexity of tailoring the model to the student.

There is one aspect of this generic student model which derives from our theory of skill acquisition and another aspect which does not. The aspect that is theoretically justified is the belief that there are not different types of students who will find different aspects of a problem differentially hard. That is, our theory does not expect individual differences in learning, beyond some overall difference in ability or motivation. The theory implies that all people learn in basically the same way. Of course, it is an open question whether there is empirical evidence for the theory on this score. In our own research it does appear that students differ only in a single dimension of how well they learn. Despite valiant searches we have yet to find evidence that one set of productions cluster together as difficult for one group of students while a different cluster of productions are difficult for another group of students.

The aspect of use of a generic model that does not derive from the theory is the assumption that past history of use with a rule implies nothing about the interpretation of a current error. We have evidence that different subjects continue to have trouble with specific different rules. (This is to be contrasted with a trait view that says there is a non-singleton set of productions that a number of subjects will have difficulty with). If the student has had a past history of success on a rule it is more likely that error reflects a slip, rather than some fundamental misunderstanding. Currently, our tutors treat all errors as if they reflected fundamental misconceptions and offers detailed explanation, but the better response sometimes would be simply to point the error out.

(3) Problem Sequence

The existing tutors implement a mastery model for controlling selection of problems to present to the student. They maintain an assessment of the student's performance on various rules and have knowledge of what problems exercise what rules. They will not let the student move on to problems involving new rules until the student is above a threshold of competence on the current rules. If the student has not demonstrated mastery, the tutor will select additional problems from the current set which exercise rules on which the student is weak.

While such a mastery policy for problem sequence may seem reasonable and there is evidence in the educational literature for its effectiveness (Bloom, 1984), it is interesting to inquire as to its underlying psychological rationalization. Why not go onto new problems while the student is weak on current knowledge and teach both the new knowledge and the old weak knowledge in context of the new problems. Fundamentally, it rests on a belief in an optimal learning load--that if we overload a student with too many things to learn, he will learn none of them well. On the other hand, students are advanced to new material at some point when further training on the old material could have improved their performance even more. So there is a countervailing assumption about diminishing returns--that at some point the gain in improving performance on old rules is not equal to gain in learning new

rules.

Our choice about exactly where to set the mastery level has been entirely *ad hoc*. In the ACT and PUPS theories working-memory load affects learning and problems pose less load as they become better learned. However, these processes are not specified in a way that enables us to define an optimal next problem. The issue of problem sequence and mastery levels remains to be worked out in a model tracing paradigm.

(4) Declarative Instruction

A student's first introduction to the knowledge required to solve a class of problems is typically not from the tutor; rather it is declarative instruction typically in textbook or lecture format. How should this declarative instruction be formulated to make it maximally helpful in learning the skill? Given our analysis of learning by analogy, instruction should take the form of examples appropriate for mapping into problem solutions. Given our PUPS structures, it is not enough that the student simply have the form slots of these structures properly represented; it is critical for successful learning that the student have properly represented the function of these structures and any prerequisites to these structures achieving their functions. For example, Pirolli and Anderson (1985) showed that, while all students learn recursive programming by analogy to existing programs, what determines how well they learn is how well they represent how these programs achieve their function. Basically, students often understand an example only superficially and thus emerge from analogy with mischaracterizations of the range of problems for which the structures in the example are appropriate.

In our efforts to create textual instruction to go along with our tutors, we have focused on the issue of giving good examples for purposes of mapping and trying to assure that the student achieves the right encoding of the example. Indeed, we are producing a LISP textbook (Anderson, Corbett, Reiser, in press) which consists mainly of carefully crafted examples with explanation aimed at promoting the right encoding.

The Interface

One might have thought that given the discussion above, the description of tutoring would be complete. We have stated how it models a student and how it uses that model to achieve pedagogical goals. However, this discussion is abstract and leaves completely unspecified what the student actually experiences, which is the computer interface. We have learned that design of the interface can make or break the effectiveness of the tutor. Below are just a few examples:

1. Early in the history of the LISP tutor we had a system in which the student entered code in a buffer and then dispatched the contents of that buffer to appear in a code window. The students were always getting confused about what code they should be entering. We changed this to a system where one typed the new code right into the old code and all of these confusions disappeared.

2. An early version of the algebra tutor had a system where students entered a next equation, the tutor figured out what steps they engaged in, and tried to give appropriate feedback and point them back to the right track. The problem was that the students' error might well have been at some intermediate step that the students were no longer fixated upon (e.g., adding two fractions as part of moving a number across the equation sign). It was very difficult to communicate to the student what the problem was. We introduced the system described earlier in this paper in which the student actually stepped through the microsteps of the transformation in a relatively painless way with the system. The tutor could flag the errors as they occurred and these miscommunications disappeared.

3. We used to have our students type in geometry statements through a typical keyboard. Given the rather special syntax of geometry statements, students would often enter basically correct statements in syntactically incorrect form. When the system told them it could not understand what they meant, they doubted their understanding of the problem and often regressed. We replaced this with a system that allowed them to use a mouse to

enter statements by pointing to a menu of geometry expressions. We also introduced a real time parser which flagged them as soon as they entered a term which would make their expression syntactically illegal. Again our difficulties disappeared.

4. The graphical structure we use to represent geometry statements (Figures 4-8) seems to be the key to enabling students to understand the structure of a proof even though it is essentially isomorphic in logical structure to a linear proof. The graphical structures make explicit the logical relationship they would have to infer.

5. In all of our tutors it seems critical to spend considerable time fashioning the English to make it as brief and as understandable as possible. If students face great masses of hard-to-understand prose, they will simply not process the message.

6. Performance on the LISP tutor improved when we introduced a facility to bring up the problem statement at any point in time, and when there is room on the screen, the problem statement is automatically displayed. Performance in the geometry tutor improved when we introduced a facillity for bringing up statements of geometry postulates at will.

7. One of the major disadvantages of all of our tutors compared to human tutors is that, at least so far, they use only the visual medium. This means that students must move their eyes from the problem to process the textual instruction. In contrast, with a human tutor, the student can listen to the tutor while continuing to look at the the problem and even have parts of the problem emphasized through the tutor's pointing.

These observations illustrate two general points about interface design for tutors:

(a) It is important to have a system that makes it clear to a student where he or she is in the problem solution and where their errors are (Observations 1, 2, and 3)

(b) It is important to minimize working memory and processing load involved in the problem solving (Observations 4, 5, 6, and 7).

While one wants an interface with these properties, it is important that the interface itself be easy to learn and use. One does not want the task of dealing with the interface to come to dominate learning the subject material. An easy interface is one that minimizes the number of things to be learned and minimizes the number of actions (e.g., keystrokes, mouseclicks) that the student has to perform to communicate to the tutor. Its learnability is enhanced if it is as congruent with past experience as possible. It should also have a structure that is as congruent as possible with the problem structure. Finally, the actions should be as internally consistent as possible.

Conclusions

What we have described is a theoretical framework for our tutoring work and some experiences based on that framework. Both the tutors and the theory are evolving objects and so it is not the case that the current embodiments of our tutors reflect all of the current insights of our theory. Still there is an approximation here, and it is worthwhile to ask to what degree has our tutoring experience been successful and to what degree has our tutoring experience confirmed the theory.

The first observation is that students do seem to learn from the tutors. We think this is quite a remarkable fact and not something that we had really believed would work so well when we set out to build these tutors. We have taken cognitive models of the information-processing, embedded them in instructional systems, and nothing has fallen apart. They can embody substantial amounts of material, can be developed in feasible time, run within acceptable bounds of efficiency, and are robust in their behavior. While we have no truly satisfactory evaluations of the tutors they seem to be better than solving problems with no tutor and students claim to enjoy working with them. This feasibility demonstration gives some credence to the general theoretical framework in which the tutors were built.

It is a separate question of whether the students behave and learn with the tutors as the theory would predict. This is a difficult question to assess because the theory is

probabilistic and does not specify in advance probabilities such as the probability of encoding a production: rather these probabilities would have to be estimated from the data. It is also difficult because the theory makes predictions only given students' encodings of the instruction and of the problem, and students clearly do vary in how they encode this information. Nonetheless, what analyses we have done do seem to confirm the theory. Figure 10 presents an analysis of some data from the LISP tutor that monitors time to type in code corresponding to the firing of individual productions. So for instance typing "(cons" corresponds to the firing of a production that recognizes the applicability of the CONS function. We have put in times associated with the firing of productions learned in lessons 1, 2, and 3. We have plotted averages for all the productions introduced in a lesson but these averages are typical of individual productions (Anderson, in press). What these times include is somewhat complex because they include a lot of low-level interactions with the tutor. However, they should reflect the learning of the production or productions associated with the execution of the code. These learning curves have just the shape we would predict. There is a big drop from first application of a production to second as the knowledge is compiled into production form and gradual speed-up after that as the production rules become strengthened.

Insert Figure 10 about here

Another observation is that the knowledge acquired does seem to have the appropriate range. For instance, in the geometry tutor students can apply new combinations of the inference rules that they have learned to solve a new problem as long as the contextual heuristics that recommend the application of these rules are ones they have already encountered. On the other hand if they are presented with a problem that can be solved by the knowledge they have been taught, but which requires applying a new contextual heuristic to see the applicability of that rule, they experience great difficulty.

References

Anderson, J.R. (1981). Tuning of search of the problem space for geometry proofs. In A. Drinan (Ed.), *Proceedings of IJCAI-81, Vol. I.* Vancouver, British Columbia: International Joint Conference on Artificial Intelligence, 165-170.

Anderson, J.R. (1982). Acquisition of proof skills in geometry. In J.G. Carbonell, R. Michalski & T. Mitchell (Eds.), *Machine Learning, An Artificial Intelligence Approach.* Palo Alto, CA: Tioga Press.

Anderson, J.R. (1982). Acquisition of Cognitive Skill. *Psychological Review* ,89, 369-406.

Anderson, J.R. (1983). *The Architecture of Cognition.* Cambridge, MA: Harvard University Press.

Anderson, J.R. (in press). Production systems, learning, and tutoring. In D. Klahr, P. Langley, & R. Neches (Eds.), *Self-modifying Production Systems: Models of Learning and Development.* Cambridge, MA: Bradford Books/MIT Press.

Anderson, J.R. (in preparation). Methodologies for studying human knowledge.

Anderson, J.R. & Jeffries, R. (1985). Novice LISP errors: Undetected losses of information from working memory. *Human-Computer Interaction* ,22, 403-423.

Anderson, J. R., Boyle, C. F., & Yost, G. (1985). The Geometry Tutor. In *Proceedings of IJCAI-85.* Los Angeles, CA: IJCAI. 1-7.

Anderson, J.R., Boyle, C.F., Farrell, R., & Reiser, B.J. (in press). Cognitive principles in the design of computer tutors. In P. Morris (Ed.), *Modelling Cognition.* Wiley.

Anderson, J.R., Corbett, A.T., & Reiser, B.J. (in press). *Essential LISP.* Reading, MA: Addison-Wesley.

Anderson, J.R., Farrell, R., & Sauers, R. (1984). Learning to program in LISP. *Cognitive Science* ,8, 87-129.

Anderson, R. C., Kulhavy, R. W., & Andre, T. (1972). Conditions under which feedback facilitates learning for programmed lessons. *Journal of Education Psychology* ,63,

186-188.

Bloom, B.S. (1984). The 2 Sigma Problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher* ,*13*, 3-16.

Brown, J.S. & Burton, R.R. (1978). Diagnostic models for procedural bugs in basic mathematical skills. *Cognitive Science* ,2, 155-192.

Brown, J.S. & Van Lehn, K. (1980). Repair theory: A generative theory of bugs in procedural skills. *Cognitive Science* ,4, 379-426.

Brown, J.S., Burton, R.R., & DeKleer, J. (1982). Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II and III. In Sleeman & Brown (Eds.), *Intelligent Tutoring Systems*. New York: Academic Press.

Carroll, J. (March 1985). *Designing minimalist training materials* (Research Report 46643). IBM Watson Research Center, Computer Science Department.

Clancey, W.J. (1982). Tutoring rules for guiding a case method dialogue. In D. Sleeman & J.S. Brown (Eds.), *Intelligent Tutoring Systems*. New York: Academic Press.

Collins, A., Warnock, E., & Passafuime, J. (1978). Analysis and synthesis of tutorial dialogues. In G. Bower (Ed.), *The Psychology of Learning and Motivation*. New York: Academic Press.

Dulany, D. E., Carlson, R. A., & Dewey, G. I. (1984). A case of syntactical learning and judgment: How conscious and how abstract? *Journal of Experimental Psychology: General* ,*113*, 541-555.

Ericsson, K. A., & Simon, H. A. (1984). *Protocol Analysis: Verbal Reports as Data*. Cambridge, MA: The MIT Press.

Forgy, C.L. (1982). Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence* ,*19*, 17-37.

Johnson, L. & Soloway, E. (1984). Intention-based diagnosis of programming errors. In *Proceedings of the National Conference on Artificial Intelligence*. Austin, TX: NCAI.

Lewis, M. W. & Anderson, J. R. (1985). Discrimination of operator schemata in problem solving: Learning from examples. *Cognitive Psychology* ,*17*, 26-65.

Lewis, M.W., Milson, R., & Anderson, J.R. (In press). Designing an intelligent authoring system for high school mathematics ICAI: The TEACHERS APPRENTICE Project. In Greg Kearsley (Ed.), *Artificial Intelligence and Instruction: Applications and Methods*. Addison-Wesley.

Matz, M. (1982). Towards a process model for high school algebra. In D. Sleeman and J.S. Brown (Eds.), *Intelligent Tutoring Systems*. New York: Academic Press.

Norman, D. A. (1981). Categorization of action slips. *Psychological Review* ,*88*, 1-15.

Pirolli, P. L. & Anderson, J. R. (1985). The role of learning from examples in the acquisition of recursive programming skill. *Canadian Journal of Psychology* ,*39*, 240-272.

Reder, L. M., Charney, D. H., & Morgan, K. I. (In press). The role of elaborations in learning a skill from an instructional text. Memory and Cognition.

Reiser, B. J., Anderson, J. R., & Farrell, R. G. (1985). Dynamic student modelling in an intelligent tutor for LISP programming. In *Proceedings of IJCAI-85*. Los Angeles, CA: IJCAI. 8–14.

Sleeman, D. (1982). Assessing aspects of competence in basic algebra. In D. Sleeman & J.S. Brown (Eds.), *Intelligent Tutoring Systems*. New York: Academic Press.

Sleeman, D. & Brown, J.S. (Eds.). (1982). *Intelligent Tutoring Systems*. New York: Academic Press.

Winograd, T. (1975). Frame representations and the declarative procedural controversy. In D. Bobrow & A. Collins (Eds.), *Representation and Understanding*. New York: Academic Press.

Winston, P. H. & Horn, B. K. P. (1984). *LISP, Second Edition*. Reading, MA: Addison-Wesley.

Table 1

Define the function fact. It takes a number argument, n and computes the factorial of n, which is the result of multiplying n x n-1 x n-2 ... x 1. For instance, (fact 3) = 3 x 2 x 1 = 6. By definition, the factorial of 0 is 1.

1. **(defun fact (n)**
   (cond ((equal) *<ACTION>)*
   <RECURSIVE-CASE>))

   I assume that you are trying to test if the number argument is equal to the number zero. There is a special function for testing equivalence to zero. Try to use that function instead of EQUAL.

   What code can you write to test if the condition for terminating case is true? Try writing a predicate.

2. *(defun fact (n)*
   (cond ((zero *<ACTION>)*
   <RECURSIVE-CASE>))

   Did you mean zercp (y or n)? y

3. *(defun fact (n)*
   (cond ((zerop **n) 1)**
   <RECURSIVE-CASE>))

   is the value of fact in the recursive cases calculated by:

PRESS:	IF YOU WANT TO:
1	Calling fact recursively with n.
2	Multiplying n by fact of n minus 1.
3	Getting 1 2 ... n and then multiplying the results.
4	have the tutor choose

   Menu choice: 3

That is a reasonable way to think of doing factorials but it is not a plan for a recursive function. Since you seem to be having trouble with the recursive cases let us work through some examples and figure out the conditions and actions for each of these cases.

The recursive relationship is the relationship between fact called with n and fact called with:

PRESS:	IF YOU WANT TO:
1	n minus 1
2	n equal to 0

3 n plus 1
4 have the tutor choose
Menu choice: 1

We will consider some concrete examples. We will characterize the recursive relationship between fact called with 1 and fact called with 0

What is the value of (fact 1)?
Answer: 1

What is the value of (fact 0)?
Answer: 1

EXAMPLES

	fact (n)	fact (n - 1)
A.	(fact 1) = 1	(fact 0) = 1

We will consider some more concrete examples. We will characterize the recursive relationship between fact called with 3 and fact called with 2

What is the value of (fact 3)?
Answer: 6

What is the value of (fact 2)?
Answer: 2

EXAMPLES

	fact (n)	fact (n - 1)
A.	(fact 1) = 1	(fact 0) = 1
B.	(fact 3) = 6	(fact 2) = 2

Since there is only one recursive case characterized by the examples you have just worked out, the condition in this case can be just coded as T. We will just focus on characterizing the action in this case.

In examples A and B what do you have to do to get the result of fact called with n?

PRESS:	IF YOU WANT TO:
1	Multiply n by one less than n.
2	Multiply n by fact of one less than n.
3	Add n to the result of fact called with one less than n.
4	have the tutor choose

Menu choice: 2

We have now worked through a series of examples that illustrate the recursive relation.

We will now return to the goal of trying to produce LISP code that specificies the recursive relation

4.
```
(defun fact (n)
    (cond ((zerop n) 1)
          (t (fact ))))
```

You will have to use fact at some point but right now you want to multiply.

5.
```
(defun fact (n)
    (cond ((zerop n) 1)
          (t (times <NUM> <NUM2>))))
```

What do you want to do first?

PRESS:	IF YOU WANT TO:
1	Write code to get the number argument
2	Write code to get the recursive part

Menu choice: 1

6.
```
(defun fact (n)
    (cond ((zerop n) 1)
          (t (times n (fact n))))).
```

You cannot just code n at this point. What you want to do is subtract one from n.

7.
```
(defun fact (n)
    (cond ((zerop n) 1)
          (t (times n (fact (difference )))))
```

Difference would work in this case but it is more appropriate to use the function SUB1 when you are subtracting 1 from a number.

8.
```
(defun fact (n)
    (cond ((zerop n) 1)
          (t (times n (fact (sub1 n))))))
```

— YOU ARE DONE. TYPE NEXT TO GO ON AFTER —
— TESTING THE FUNCTIONS YOU HAVE DEFINED —

THE LISP WINDOW

```
=>(trace fact)
(fact)
=> (fact 3)
1 <Enter> fact (3)
|2 <Enter> fact (2)
| 3 <Enter> fact (1)
| |4 <Enter> fact (0)
```

```
| |4 <EXIT>  fact  1
| 3 <EXIT>  fact  1
|2 <EXIT>  fact  2
1 <EXIT>  fact  6
6
```

Figure Captions

Figure 1 — A problem that occurs early in the problem sequence used with the geometry tutor.

Figure 2 — The screen configuration before line 4 in Table 1.

Figure 3 — The screen configuration at the end of the dialogue in Table 1.

Figure 4 — An initial screen configuration with the geometry tutor.

Figure 5 — The screen configuration after the student has selected the premises and the rule and is about to enter the conclusion.

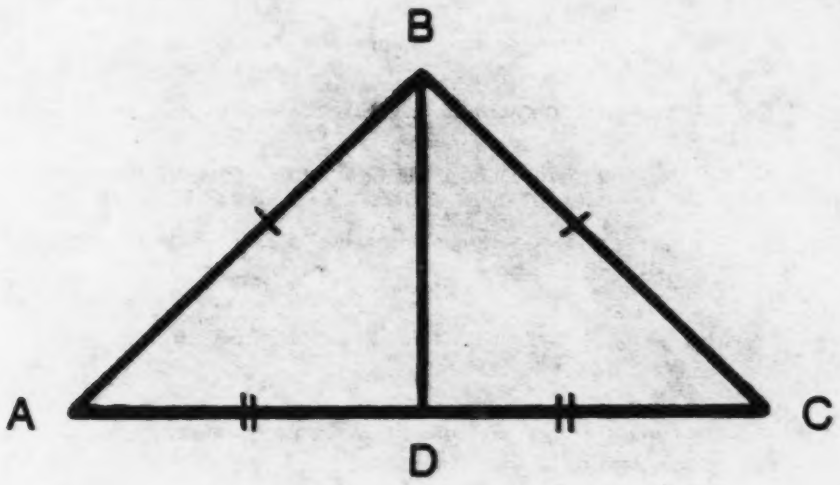
Figure 6 — The student has just tried to apply ASA to the two premises $\angle EJX \cong \angle EJY$, $\angle EXJ \cong \angle EXK$

Figure 7 — The student has succeeded in proving one of the two requisite triangle congruences.

Figure 8 — The proof of the problem is now complete.

Figure 9 — The algebra tutor's interface. The tutoring window is at the top. Below is the current equation. The notebook below keeps a history of the problem solution.

Figure 10 — Plot of learning data from the LISP tutor. Time is plotted to code symbol corresponding to the firing of productions that were introduced in the first, second, and third lessons.



Given: $\overline{AB} \cong \overline{BC}$

$\overline{AD} \cong \overline{CD}$

Prove: $\triangle ABD \cong \triangle CBD$

Fig. 1

In examples A and B what do you have to do to get the result of fact called with n?

PRESS: IF YOU WANT TO:

1. Multiply n by one less than n.
2. Multiply n by fact of one less than n.
3. Add n to the result of fact called with one less than n.
4. Have the tutor choose.

Menu Choice: 2

CODE FOR fact

```
(defun fact (n)
    (cond ((zerop n) 1)
          <RECURSIVE-CASE>))
```

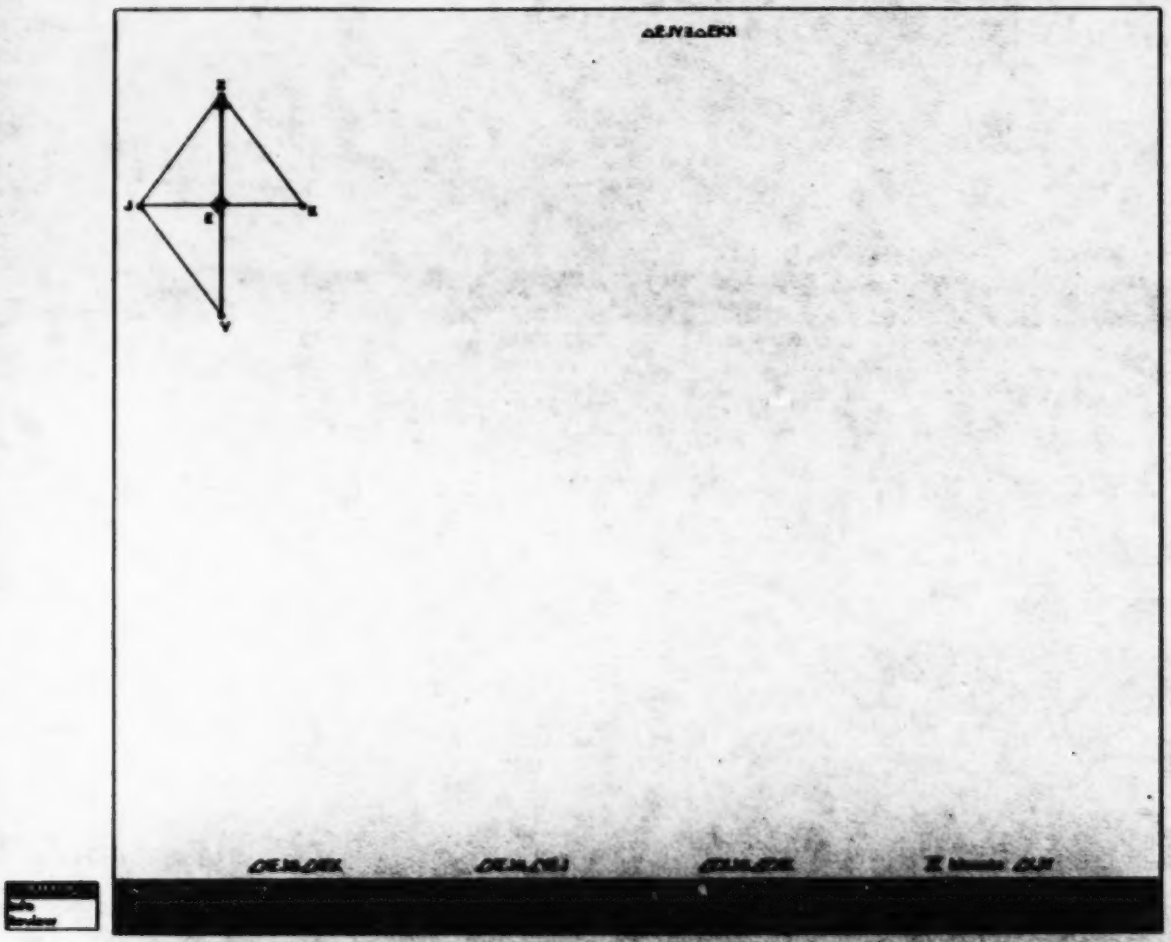
EXAMPLES

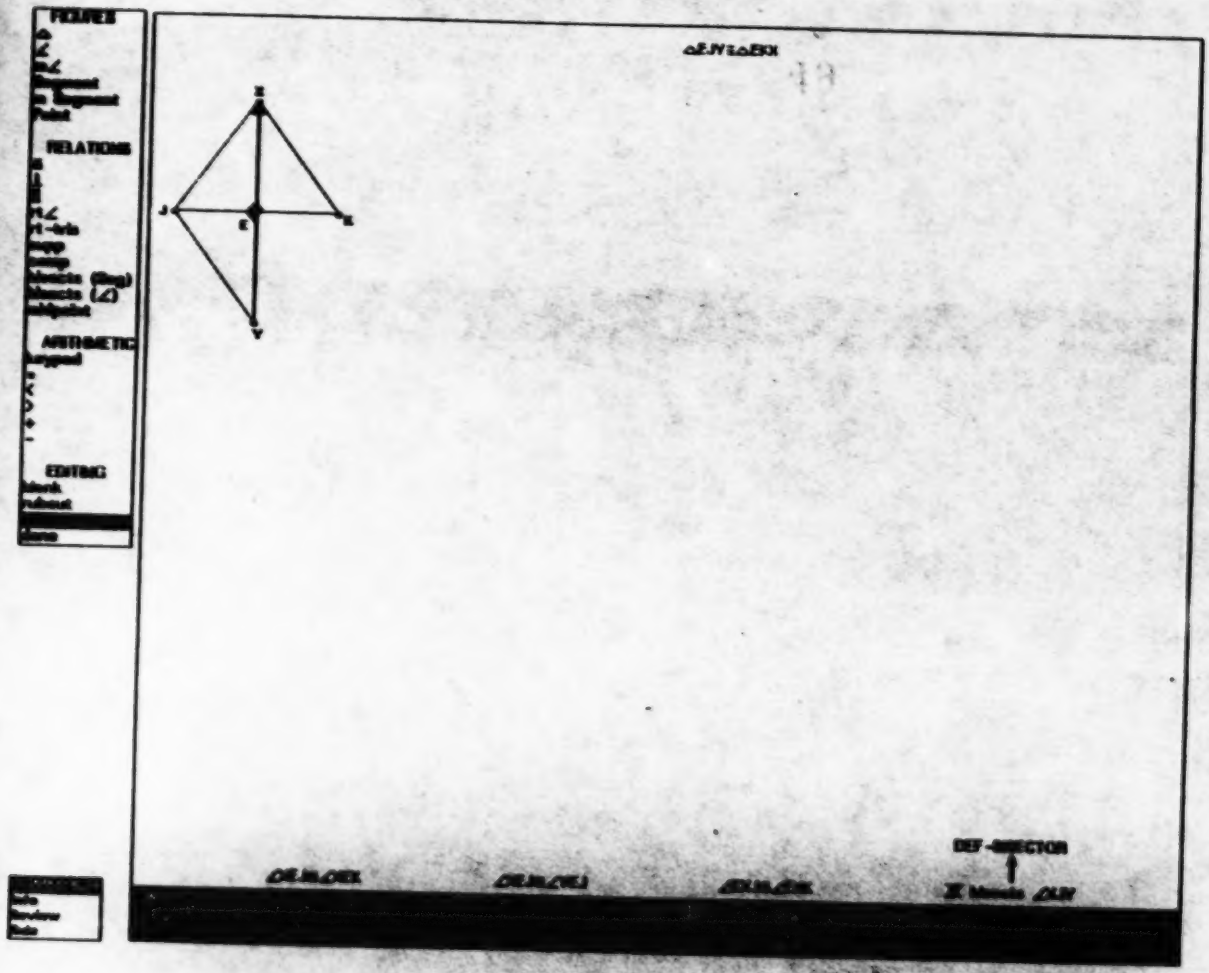
	fact (n)	fact (n-1)
A.	(fact 1) = 1	(fact 0) = 1
B.	(fact 3) = 6	(fact 2) = 2

```
-- YOU ARE DONE.  TYPE NEXT TO GO ON AFTER --
-- TESTING THE FUNCTIONS YOU HAVE DEFINED  --

(defun fact (n)
    (cond ((zerop n) 1)
          (t (times n (fact (sub1 n))))))
```

THE LISP WINDOW

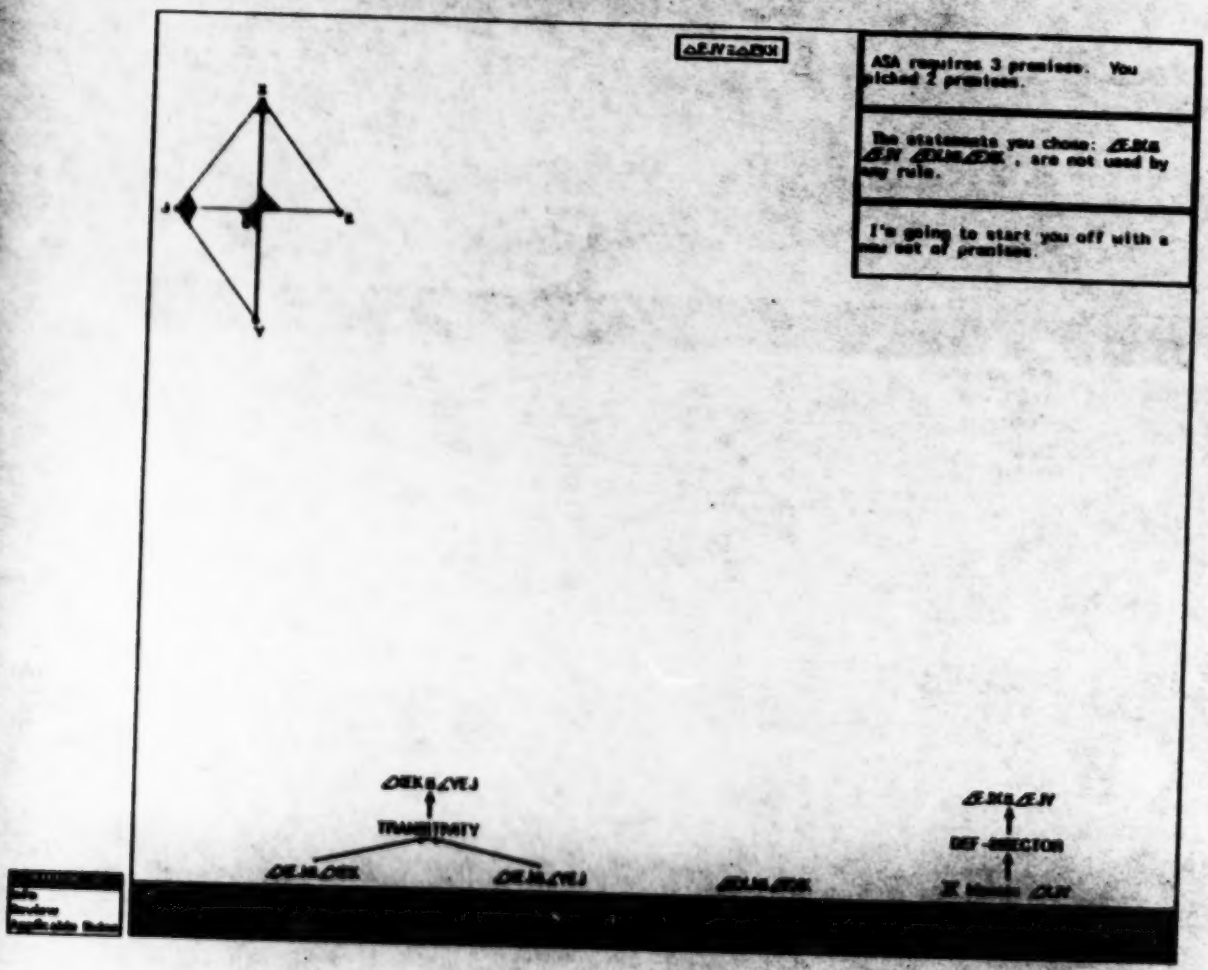
```
=> (trace fact)
(fact)

=> (fact 3)
1 <Enter> fact (3)
|2 <Enter> fact (2)
| 3 <Enter> fact (1)
| |4 <Enter> fact (0)
| |4 <EXIT>  fact  1
| 3 <EXIT>  fact  1
|2 <EXIT>  fact  2
1 <EXIT>  fact  6
6
```

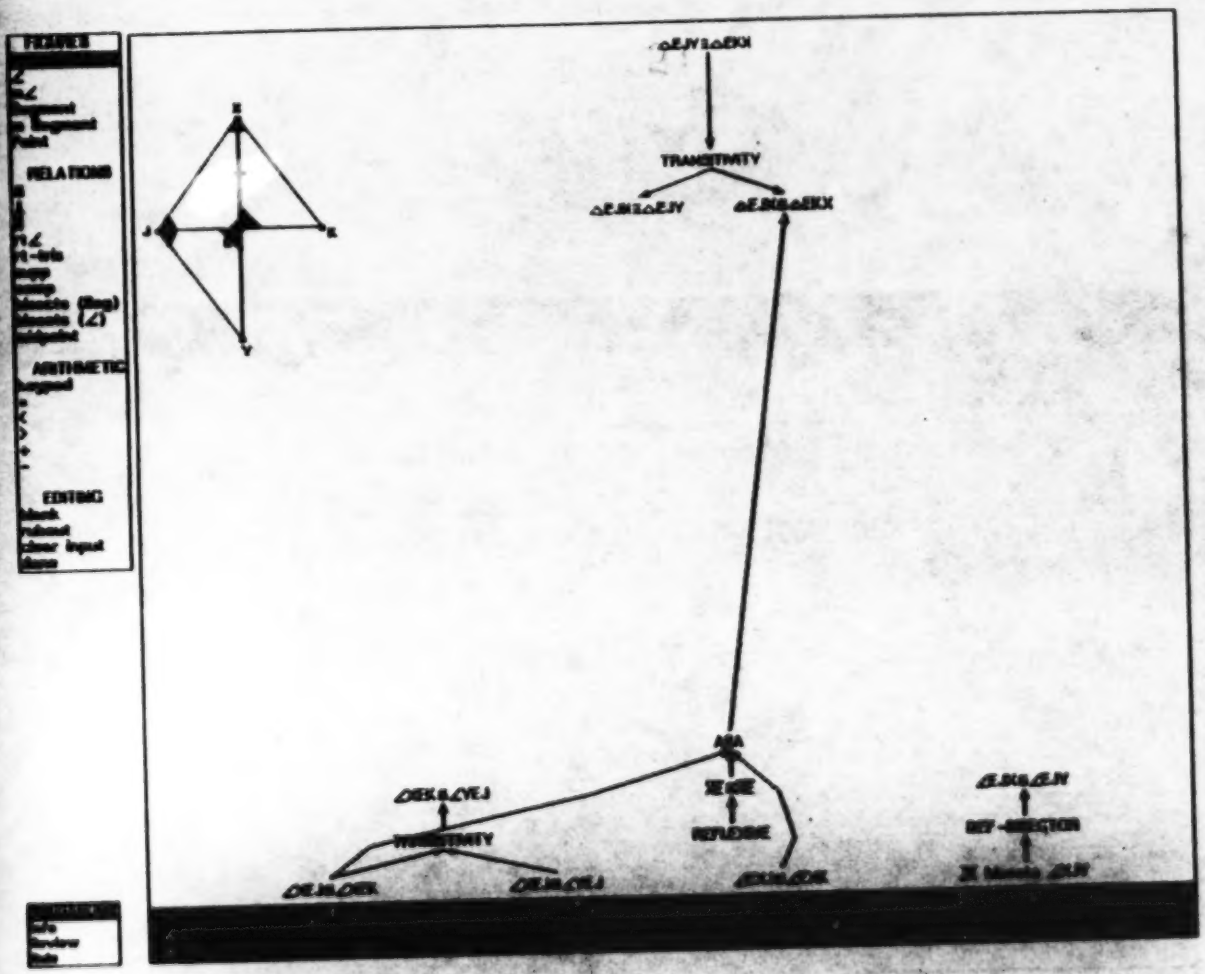




BEST COPY AVAILABLE

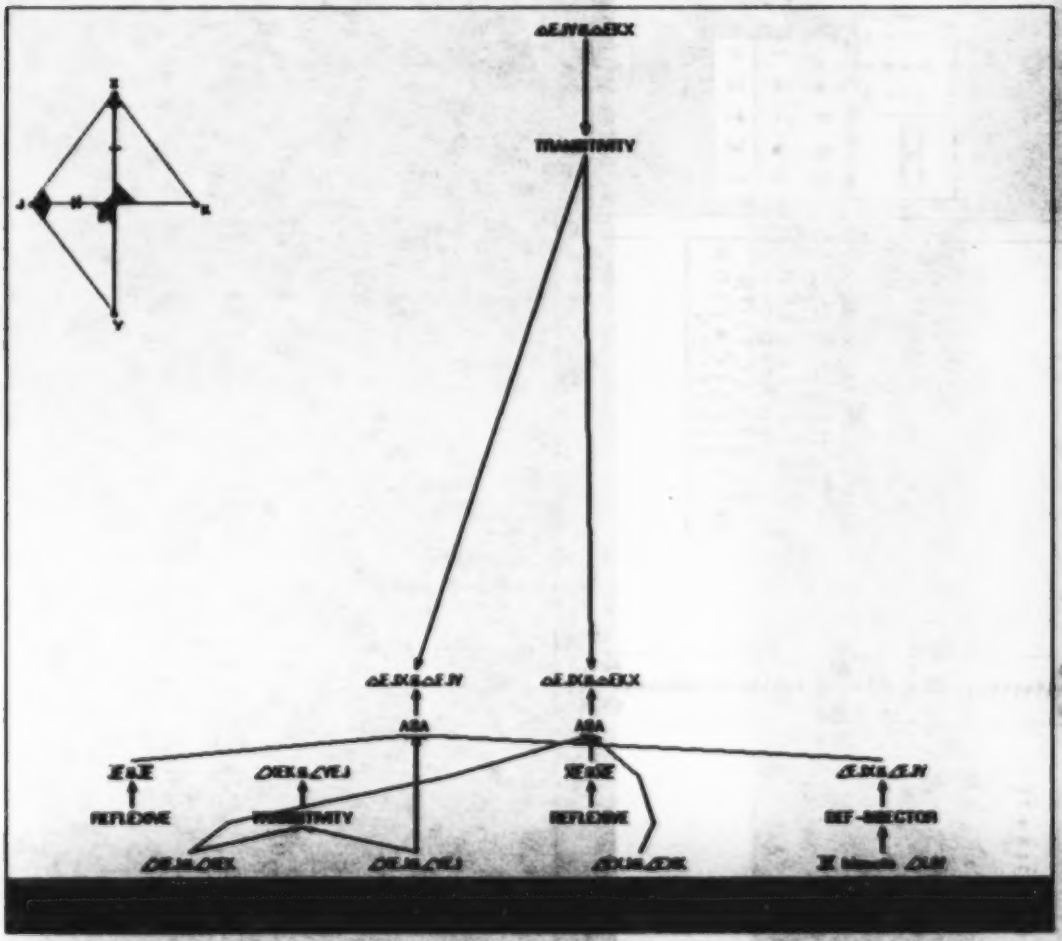
BEST COPY AVAILABLE


ASA requires 3 premises. You picked 2 premises.
The statements you chose: , are not used by any rule.
I'm going to start you off with a new set of premises.
TRANSITIVITY
DEF-DIRECTION



BEST COPY AVAILABLE

BEST COPY AVAILABLE


TRANSITIVITY
ABA
ABA
REFLEXIVE
REFLEXIVE
REF.-SELECTOR

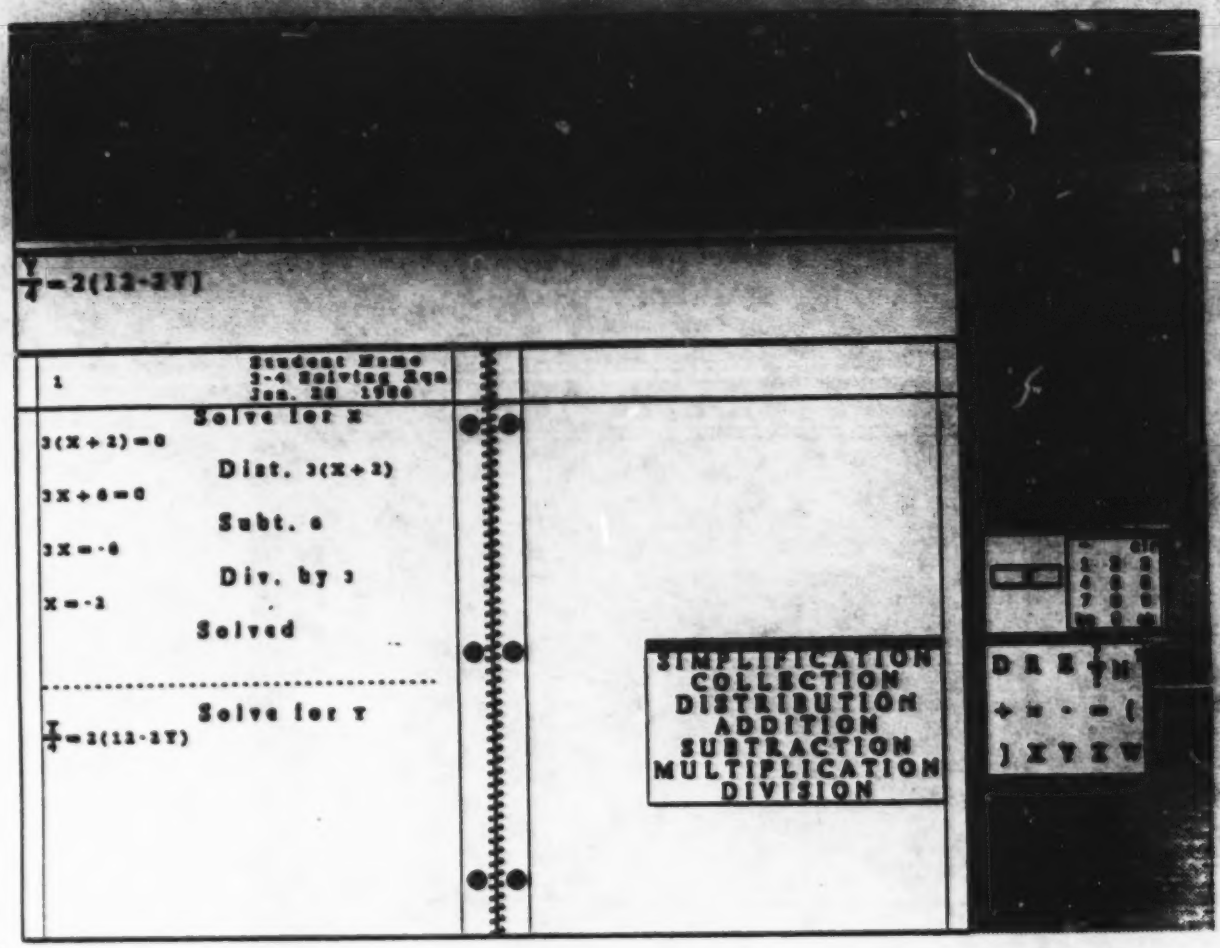

Student Name
3-4 Solving Eqn
Jan. 28 1980
Solve for X
3(X+2)=0
Dist. 3(X+2)
3X+6=0
Subt. 6
3X=-6
Div. by 3
X=-2
Solved
Solve for Y
SIMPLIFICATION
COLLECTION
DISTRIBUTION
ADDITION
SUBTRACTION
MULTIPLICATION
DIVISION

FIGURE 9

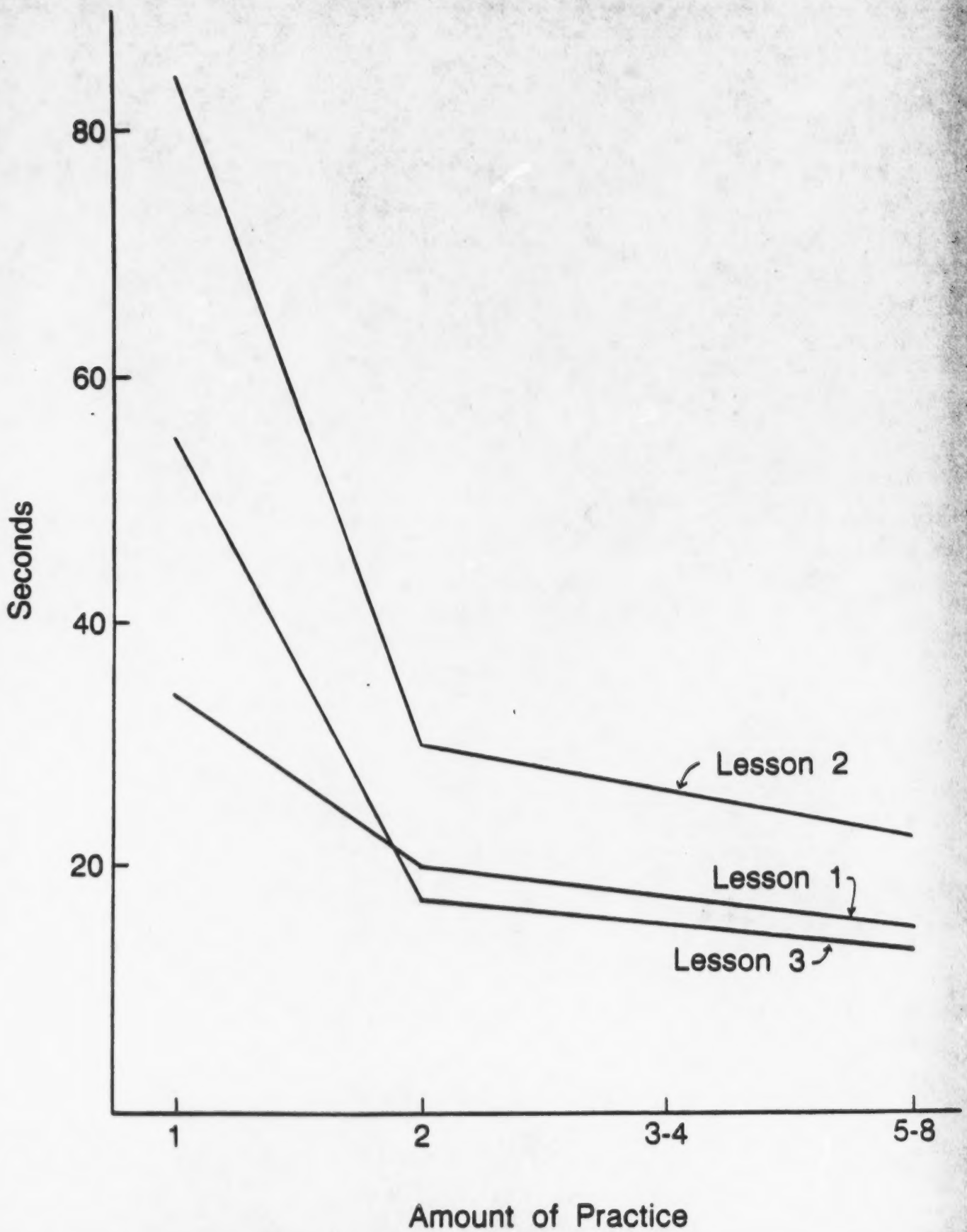

80
60
40
20
Seconds
Lesson 2
Lesson 1
Lesson 3
1
2
3-4
5-8
Amount of Practice

Fig. 10

Air Force Human Resources Lab
AFHRL/MPD
Brooks AFB, TX 78235

AFOSR,
Life Sciences Directorate
Bolling Air Force Base
Washington, DC 20332

Dr. Robert Ahlers
Code N711
Human Factors Laboratory
NAVTRAEQUIPCEN
Orlando, FL 32813

Dr. Ed Aiken
Navy Personnel R&D Center
San Diego, CA 92152

Dr. James Anderson
Brown University
Center for Neural Science
Providence, RI 02912

Dr. John R. Anderson
Department of Psychology
Carnegie-Mellon University
Pittsburgh, PA 15213

Dr. Nancy S. Anderson
Department of Psychology
University of Maryland
College Park, MD 20742

Technical Director, ARI
5001 Eisenhower Avenue
Alexandria, VA 22333

Special Assistant for Projects,
OASN(M&RA)
5D800, The Pentagon
Washington, DC 20350

Dr. Alan Baddeley
Medical Research Council
Applied Psychology Unit
15 Chaucer Road
Cambridge CB2 2EF
ENGLAND

Dr. Patricia Baggett
University of Colorado
Department of Psychology
Box 345
Boulder, CO 80309

Dr. Eva L. Baker
UCLA Center for the Study
of Evaluation
145 Moore Hall
University of California
Los Angeles, CA 90024

Dr. Meryl S. Baker
Navy Personnel R&D Center
San Diego, CA 92152

Dr. William M. Bart
University of Minnesota
Dept. of Educ. Psychology
330 Burton Hall
178 Pillsbury Dr., S.E.
Minneapolis, MN 55455

Dr. Menucha Birenbaum
School of Education
Tel Aviv University
Tel Aviv, Ramat Aviv 69978
ISRAEL

Dr. Gautam Biswas
Department of Computer Science
University of South Carolina
Columbia, SC 29208

Dr. John Black
Yale University
Box 11A, Yale Station
New Haven, CT 06520

Dr. Arthur S. Blaiwes
Code N711
Naval Training Equipment Center
Orlando, FL 32813

Dr. Robert Blanchard
Navy Personnel R&D Center
San Diego, CA 92152

Cdt. Arnold Bohrer
Sectie Psychologisch Onderzoek
Rekruterings-En Selectiecentrum
Kwartier Koningen Astrid
Bruijnstraat
1120 Brussels, BELGIUM

Dr. Jeff Bonar
Learning R&D Center
University of Pittsburgh
Pittsburgh, PA 15260

Dr. Gordon H. Bower
Department of Psychology
Stanford University
Stanford, CA 94306

Dr. Robert Breaux
Code N-095R
NAVTRAEQUIPCEN
Orlando, FL 32813

Dr. Ann Brown
Center for the Study of Reading
University of Illinois
51 Gerty Drive
Champaign, IL 61280

Dr. John S. Brown
XEROX Palo Alto Research
Center
3333 Coyote Road
Palo Alto, CA 94304

Dr. Bruce Buchanan
Computer Science Department
Stanford University
Stanford, CA 94305

Maj. Hugh Burns
AFHRL/IDE
Lowry AFB, CO 80230-5000

Dr. Patricia A. Butler
NIE Mail Stop 1806
1200 19th St., NW
Washington, DC 20208

Dr. Jaime Carbonell
Carnegie-Mellon University
Department of Psychology
Pittsburgh, PA 15213

Dr. Susan Carey
Harvard Graduate School of
Education
337 Gutman Library
Appian Way
Cambridge, MA 02138

Dr. Pat Carpenter
Carnegie-Mellon University
Department of Psychology
Pittsburgh, PA 15213

Dr. Robert Carroll
NAVOP 0187
Washington, DC 20370

Chair, Department of
Psychology
College of Arts and Sciences
Catholic University of
America
Washington, DC 20064

Dr. Fred Chang
Navy Personnel R&D Center
Code 51
San Diego, CA 92152

Dr. Davida Charney
Department of Psychology
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213

Dr. Eugene Charniak
Brown University
Computer Science Department
Providence, RI 02912

Dr. Michelene Chi
Learning R & D Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15213

Mr. Raymond E. Christal
AFHRL/MOE
Brooks AFB, TX 78235

Professor Chu Tien-Chen
Mathematics Department
National Taiwan University
Taipei, TAIWAN

Dr. William Clancey
Stanford University
Knowledge Systems Laboratory
701 Welch Road, Bldg. C
Palo Alto, CA 94304

Dr. David E. Clement
Department of Psychology
University of South Carolina
Columbia, SC 29208

Dr. Allan M. Collins
Bolt Beranek & Newman, Inc.
50 Moulton Street
Cambridge, MA 02138

Dr. John J. Collins
Director, Field Research
Office, Orlando
NPRDC Liaison Officer
NTSC Orlando, FL 32813

Dr. Stanley Collyer
Office of Naval Technology
Code 222
800 N. Quincy Street
Arlington, VA 22217-5000

Dr. Lynn A. Cooper
Learning R&D Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15213

CAPT P. Michael Curran
Office of Naval Research
800 N. Quincy St.
Code 125
Arlington, VA 22217-5000

Dr. Cary Czichon
Mail Station 3407
Texas Instruments AI Lab
P.O. Box 405
Lewisville, TX 75067

Bryan Dallman
AFHRL/LRT
Lowry AFB, CO 80230

Dr. Natalie Dehn
Department of Computer and
Information Science
University of Oregon
Eugene, OR 97403

Dr. Sharon Derry
Florida State University
Department of Psychology
Tallahassee, FL 32306

Dr. R. K. Dismukes
Associate Director for Life Sciences
AFOSR
Boiling AFB
Washington, DC 20332

Defense Technical
Information Center
Cameron Station, Bldg 5
Alexandria, VA 22314
Attn: TC
(12 Copies)

Dr. Thomas M. Duffy
Communications Design Center
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213

Dr. Richard Duran
University of California
Santa Barbara, CA 93106

Dr. John Ellis
Navy Personnel R&D Center
San Diego, CA 92252

Dr. Jeffrey Elman
University of California,
San Diego
Department of Linguistics, C-008
La Jolla, CA 92093

Dr. Richard Elster
Deputy Assistant Secretary
of the Navy (Manpower)
OASN (M&RA)
Department of the Navy
Washington, DC 20350-1000

Dr. Susan Embretson
University of Kansas
Psychology Department
Lawrence, KS 66045

Dr. Randy Engle
Department of Psychology
University of South Carolina
Columbia, SC 29208

Dr. William Epstein
University of Wisconsin
W. J. Brogden Psychology Bldg.
1202 W. Johnson Street
Madison, WI 53706

ERIC Facility-Acquisitions
4833 Rugby Avenue
Bethesda, MD 20014

Dr. K. Anders Ericsson
University of Colorado
Department of Psychology
Boulder, CO 80309

Dr. Edward Esty
Department of Education, OERI
Room 717D
1200 19th St., NW
Washington, DC 20208

Dr. Jean Claude Falmagne
Department of Psychology
New York University
New York, NY 10003

Dr. Martha Farah
Department of Psychology
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213

Dr. Beatrice J. Farr
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Marshall J. Farr
2520 North Vernon Street
Arlington, VA 22207

Dr. Pat Federico
Code 511
NPRDC
San Diego, CA 92152

Dr. Jerome A. Feldman
University of Rochester
Computer Science Department
Rochester, NY 14627

Dr. Paul Feltovich
Southern Illinois University
School of Medicine
Medical Education Department
P.O. Box 3926
Springfield, IL 62708

Mr. Wallace Feurzeig
Educational Technology
Bolt Beranek & Newman
10 Moulton St.
Cambridge, MA 02238

Dr. Craig I. Fields
ARPA
1400 Wilson Blvd.
Arlington, VA 22209

Dr. Gerhard Fischer
University of Colorado
Department of Computer Science
Boulder, CO 80309

J. D. Fletcher
9931 Corsica Street
Vienna VA 22180

Dr. Linda Flower
Carnegie-Mellon University
Department of English
Pittsburgh, PA 15213

Dr. Kenneth D. Forbus
University of Illinois
Department of Computer Science
1304 West Springfield Avenue
Urbana, IL 61801

Dr. Barbara A. Fox
University of Colorado
Department of Linguistics
Boulder, CO 80309

Dr. Carl H. Frederiksen
McGill University
3700 McTavish Street
Montreal, Quebec H3A 1Y2
CANADA

Dr. John R. Frederiksen
Bolt Beranek & Newman
50 Moulton Street
Cambridge, MA 02138

Dr. Norman Frederiksen
Educational Testing Service
Princeton, NJ 08541

Dr. Alfred R. Fregly
AFOSR/NL
Bolling AFB, DC 20332

Dr. Robert M. Gagne
1456 Mitchell Avenue
Tallahassee, FL 32303

Dr. R. Edward Geiselman
Department of Psychology
University of California
Los Angeles, CA 90024

Dr. Michael Genesereth
Stanford University
Computer Science Department
Stanford, CA 94305

Dr. Dedre Gentner
University of Illinois
Department of Psychology
603 E. Daniel St.
Champaign, IL 61820

Dr. Don Gentner
Center for Human
Information Processing
University of California
La Jolla, CA 92093

Dr. Robert Glaser
Learning Research
& Development Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15260

Dr. Arthur M. Glenberg
University of Wisconsin
W. J. Brogden Psychology Bldg.
1202 W. Johnson Street
Madison, WI 53706

Dr. Sam Glucksberg
Princeton University
Department of Psychology
Green Hall
Princeton, NJ 08540

Dr. Susan Goldman
University of California
Santa Barbara, CA 93106

Dr. Sherrie Gott
AFHRL/MODJ
Brooks AFB, TX 78235

Dr. T. Govindaraj
Georgia Institute of Technology
School of Industrial & Systems
Engineering
Atlanta, GA 30332

Jordan Grafman, Ph.D.
Department of Clinical
Investigation
Walter Reed Army Medical Center
6825 Georgia Ave., N. W.
Washington, DC 20307-5001

Dr. Richard H. Granger
Department of Computer Science
University of California, Irvine
Irvine, CA 92717

Dr. Wayne Gray
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. James G. Greeno
University of California
Berkeley, CA 94720

Dr. Henry M. Halff
Halff Resources, Inc.
4918 33rd Road, North
Arlington, VA 22207

Dr. Nancy F. Halff
Halff Resources, Inc.
4918 33rd Road, North
Arlington, VA 22207

Dr. Cheryl Hamel
NTSC
Orlando, FL 32813

Dr. Ray Hannapel
Scientific and Engineering
Personnel and Education
National Science Foundation
Washington, DC 20550

Stevan Harnad
Editor, The Behavioral and
Brain Sciences
20 Nassau Street, Suite 240
Princeton, NJ 08540

Janice Hart
Office of the Chief
of Naval Operations
OP-11HD
Department of the Navy
Washington, D.C. 20350-2000

Mr. William Hartung
PEAM Product Manager
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Wayne Harvey
SRI International
333 Ravenswood Ave.
Room B-S324
Menlo Park, CA 94025

Dr. Reid Hastie
Northwestern University
Department of Psychology
Evanston, IL 60201

Prof. John R. Hayes
Carnegie-Mellon University
Department of Psychology
Schenley Park
Pittsburgh, PA 15213

Dr. Barbara Hayes-Roth
Department of Computer Science
Stanford University
Stanford, CA 95305

Dr. Frederick Hayes-Roth
Teknowledge
525 University Ave.
Palo Alto, CA 94301

Dr. Shirley Brice Heath
School of Education
Stanford University
Stanford, CA 94305

Dr. Joan I. Heller
505 Haddon Road
Oakland, CA 94606

Dr. Geoffrey Hinton
Carnegie-Mellon University
Computer Science Department
Pittsburgh, PA 15213

Dr. Jim Hollan
Intelligent Systems Group
Institute for
Cognitive Science (C-015)
UCSD
La Jolla, CA 92093

Dr. Melissa Holland
Army Research Institute for the
Behavioral and Social Sciences
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Keith Holyoak
University of Michigan
Human Performance Center
330 Packard Road
Ann Arbor, MI 48109

Prof. Lutz F. Hornke
Universitat Dusseldorf
Erziehungswissenschaftliches
Universitatsstr. 1
Dusseldorf 1
WEST GERMANY

Mr. Dick Hoshaw
NAVOP-135
Arlington Annex
Room 2834
Washington, DC 20350

Dr. Ed Hutchins
Intelligent Systems Group
Institute for
Cognitive Science (C-015)
UCSD
La Jolla, CA 92093

Dr. Dillon Inouye
WICAT Education Institute
Provo, UT 84057

Dr. Alice Isen
Department of Psychology
University of Maryland
Catonsville, MD 21228

Dr. Zachary Jacobson
Bureau of Management Consulting
365 Laurier Avenue West
Ottawa, Ontario K1A 0S5
CANADA

Dr. Robert Jannarone
Department of Psychology
University of South Carolina
Columbia, SC 29208

Dr. Robert J. Jannarone
University of South Carolina
Department of Psychology
Columbia, SC 29208

Dr. Claude Janvier
Directeur, CIRADE
Universite' du Quebec a Montreal
Montreal, Quebec H3C 3P8
CANADA

COL Dennis W. Jarvi
Commander
AFHRL
Brooks AFB, TX 78235-5601

Dr. Robin Jeffries
Computer Research Center
Hewlett-Packard Laboratories
1501 Page Mill Road
Palo Alto, CA 94304

Margaret Jerome
c/o Dr. Peter Chandler
83, The Drive
Hove
Sussex
UNITED KINGDOM

Chair, Department of
Psychology
The Johns Hopkins University
Baltimore, MD 21218

Dr. Joseph E. Johnson
Assistant Dean for
Graduate Studies
College of Science and Mathematics
University of South Carolina
Columbia, SC 29208

Col. Dominique Jouslin de Noray
Etat-Major de l'Armee de Terre
Centre de Relations Humaines
3 Avenue Octave Greard
75007 Paris
FRANCE

Dr. Marcel Just
Carnegie-Mellon University
Department of Psychology
Schenley Park
Pittsburgh, PA 15213

Dr. Daniel Kahneman
The University of British Columbia
Department of Psychology
#154-2053 Main Mall
Vancouver, British Columbia
CANADA V6T 1Y7

Dr. Milton S. Katz
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Wendy Kellogg
IBM T. J. Watson Research Ctr.
P.O. Box 218
Yorktown Heights, NY 10598

Dr. Dennis Kibler
University of California
Department of Information
and Computer Science
Irvine, CA 92717

Dr. David Kieras
University of Michigan
Technical Communication
College of Engineering
1223 E. Engineering Building
Ann Arbor, MI 48109

Dr. Peter Kincaid
Training Analysis
& Evaluation Group
Department of the Navy
Orlando, FL 32813

Dr. Walter Kintsch
Department of Psychology
University of Colorado
Campus Box 345
Boulder, CO 80302

Dr. David Klahr
Carnegie-Mellon University
Department of Psychology
Schenley Park
Pittsburgh, PA 15213

Dr. Mazie Knerr
Program Manager
Training Research Division
HumRRO
1100 S. Washington
Alexandria, VA 22314

Dr. Janet L. Kolodner
Georgia Institute of Technology
School of Information
& Computer Science
Atlanta, GA 30332

Dr. Stephen Kosslyn
Harvard University
1236 William James Hall
33 Kirkland St.
Cambridge, MA 02138

Dr. Kenneth Kotovsky
Department of Psychology
Community College of
Allegheny County
800 Allegheny Avenue
Pittsburgh, PA 15233

Dr. David H. Krantz
2 Washington Square Village
Apt. # 15J
New York, NY 10012

Dr. Benjamin Kuipers
University of Texas at Austin
Department of Computer Sciences
T.S. Painter Hall 3.28
Austin, Texas 78712

Dr. David R. Lambert
Naval Ocean Systems Center
Code 441T
271 Catalina Boulevard
San Diego, CA 92152

Dr. Pat Langley
University of California
Department of Information
and Computer Science
Irvine, CA 92717

Dr. Jill Larkin
Carnegie-Mellon University
Department of Psychology
Pittsburgh, PA 15213

Dr. Jean Lave
School of Social Sciences
University of California
Irvine, CA 92717

Dr. Robert Lawler
Information Sciences, FRL
GTE Laboratories, Inc.
40 Sylvan Road
Waltham, MA 02254

Dr. Alan M. Lesgold
Learning R&D Center
University of Pittsburgh
Pittsburgh, PA 15260

Dr. Alan Leshner
Deputy Division Director
Behavioral and Neural Sciences
National Science Foundation
1800 G Street
Washington, DC 20550

Dr. Jim Levin
University of California
Laboratory for Comparative
Human Cognition
D003A
La Jolla, CA 92093

Dr. Michael Levine
Educational Psychology
210 Education Bldg.
University of Illinois
Champaign, IL 61801

Dr. Clayton Lewis
University of Colorado
Department of Computer Science
Campus Box 430
Boulder, CO 80309

Dr. Marcia C. Linn
Lawrence Hall of Science
University of California
Berkeley, CA 94720

Dr. R. Duncan Luce
Department of Psychology
Harvard University
Cambridge, MA 02138

Dr. Don Lyon
P. O. Box 44
Higley, AZ 85236

Vern Malec
NPRDC, Code P-306
San Diego, CA 92152

Dr. Jane Malin
Mail Code SR 111
NASA Johnson Space Center
Houston, TX 77058

Dr. William L. Maloy
Chief of Naval Education
and Training
Naval Air Station
Pensacola, FL 32508

Dr. Sandra P. Marshall
Dept. of Psychology
San Diego State University
San Diego, CA 92182

Dr. Manton M. Matthews
Department of Computer Science
University of South Carolina
Columbia, SC 29208

Dr. Richard E. Mayer
Department of Psychology
University of California
Santa Barbara, CA 93106

Dr. James McBride
Psychological Corporation
c/o Harcourt, Brace,
Javanovich Inc.
1250 West 6th Street
San Diego, CA 92101

Dr. Jay McClelland
Department of Psychology
Carnegie-Mellon University
Pittsburgh, PA 15213

Dr. James McMichael
Assistant for MPT Research,
Development, and Studies
NAVOP 01B7
Washington, DC 20370

Dr. Barbara Means
Human Resources
Research Organization
1100 South Washington
Alexandria, VA 22314

Dr. Arthur Melmed
U. S. Department of Education
724 Brown
Washington, DC 20208

Dr. Jose Mestre
University of Massachusetts
301 Goodell Building
Amherst, MA

Dr. Al Meyrowitz
Office of Naval Research
Code 1133
800 N. Quincy
Arlington, VA 22217-5000

Dr. Ryszard S. Michalski
University of Illinois
Department of Computer Science
1304 West Springfield Avenue
Urbana, IL 61801

Prof. D. Michie
The Turing Institute
36 North Hanover Street
Glasgow G1 2AD, Scotland
UNITED KINGDOM

Dr. George A. Miller
Department of Psychology
Green Hall
Princeton University
Princeton, NJ 08540

Dr. Mark Miller
Computer*Thought Corporation
1721 West Plano Parkway
Plano, TX 75075

Dr. Andrew R. Molnar
Scientific and Engineering
Personnel and Education
National Science Foundation
Washington, DC 20550

Dr. William Montague
NPRDC Code 13
San Diego, CA 92152

Dr. Tom Moran
Xerox PARC
3333 Coyote Hill Road
Palo Alto, CA 94304

Dr. Allen Munro
Behavioral Technology
Laboratories - USC
1845 S. Elena Ave., 4th Floor
Redondo Beach, CA 90277

Chair, Department of
Computer Science
U.S. Naval Academy
Annapolis, MD 21402

Dr. Allen Newell
Department of Psychology
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213

Dr. Mary Jo Nissen
University of Minnesota
N218 Elliott Hall
Minneapolis, MN 55455

Dr. Donald A. Norman
Institute for Cognitive Science
University of California
La Jolla, CA 92093

Director, Training Laboratory,
NPRDC (Code 05)
San Diego, CA 92152

Director, Manpower and Personnel
Laboratory,
NPRDC (Code 06)
San Diego, CA 92152

Director, Human Factors
& Organizational Systems Lab,
NPRDC (Code 07)
San Diego, CA 92152

Library, NPRDC
Code P201L
San Diego, CA 92152

Dr. Harry F. O'Neil, Jr.
University of Southern California
School of Education -- WPH 801
Dept. of Educational
Psychology and Technology
Los Angeles, CA 90089-0031

Dr. Stellan Ohlsson
Learning R & D Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15213

Director, Technology Programs,
Office of Naval Research
Code 12
800 North Quincy Street
Arlington, VA 22217-5000

Office of Naval Research,
Code 1133
800 N. Quincy Street
Arlington, VA 22217-5000

Office of Naval Research,
Code 1142
800 N. Quincy St.
Arlington, VA 22217-5000

Office of Naval Research,
Code 1142EP
800 N. Quincy Street
Arlington, VA 22217-5000

Office of Naval Research,
Code 1142PT
800 N. Quincy Street
Arlington, VA 22217-5000
(6 Copies)

Psychologist
Office of Naval Research
Branch Office, London
Box 39
FPO New York, NY 09510

Special Assistant for Marine
Corps Matters,
ONR Code 00MC
800 N. Quincy St.
Arlington, VA 22217-5000

Psychologist
Office of Naval Research
Liaison Office, Far East
APO San Francisco, CA 96503

Dr. Judith Orasanu
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Roy Pea
Bank Street College of
Education
610 W. 112th Street
New York, NY 10025

Dr. Nancy Pennington
University of Chicago
Graduate School of Business
1101 E. 58th St.
Chicago, IL 60637

Dr. David N. Perkins
Educational Technology Center
337 Gutman Library
Appian Way
Cambridge, MA 02138

Department of Computer Science,
Naval Postgraduate School
Monterey, CA 93940

Dr. Steven Pinker
Department of Psychology
E10-018
M.I.T.
Cambridge, MA 02139

Dr. Tjeerd Plomp
Twente University of Technology
Department of Education
P.O. Box 217
7500 AE ENSCHEDE
THE NETHERLANDS

Dr. Martha Polson
Department of Psychology
Campus Box 346
University of Colorado
Boulder, CO 80309

Dr. Peter Polson
University of Colorado
Department of Psychology
Boulder, CO 80309

Dr. Steven E. Poltrock
MCC
9430 Research Blvd.
Echelon Bldg #1
Austin, TX 78759-6509

Dr. Harry E. Pople
University of Pittsburgh
Decision Systems Laboratory
1360 Scaife Hall
Pittsburgh, PA 15261

Dr. Mike Posner
University of Oregon
Department of Psychology
Eugene, OR 97403

Dr. Sukai Prom-Jackson
1421 Massachusetts Ave., NW
#602
Washington, DC 20005

Dr. Joseph Psotka
ATTN: PERI-1C
Army Research Institute
5001 Eisenhower Ave.
Alexandria, VA 22333

Dr. Lynne Reder
Department of Psychology
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213

Dr. James A. Reggia
University of Maryland
School of Medicine
Department of Neurology
22 South Greene Street
Baltimore, MD 21201

Dr. Fred Reif
Physics Department
University of California
Berkeley, CA 94720

Dr. Lauren Resnick
Learning R & D Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15213

Dr. Mary S. Riley
Program in Cognitive Science
Center for Human Information
Processing
University of California
La Jolla, CA 92093

Dr. Edwina L. Rissland
Dept. of Computer and
Information Science
University of Massachusetts
Amherst, MA 01003

Dr. Paul R. Rosenbaum
Educational Testing Service
Princeton, NJ 08541

Dr. William B. Rouse
Georgia Institute of Technology
School of Industrial & Systems
Engineering
Atlanta, GA 30332

Dr. David Rumelhart
Center for Human
Information Processing
Univ. of California
La Jolla, CA 92093

Dr. Robert Sasmor
Army Research Institute
5001 Eisenhowdr Avenue
Alexandria, VA 22333

Dr. Roger Schank
Yale University
Computer Science Department
P.O. Box 2158
New Haven, CT 06520

Dr. Walter Schneider
Learning R&D Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15260

Dr. Alan H. Schoenfeld
University of California
Department of Education
Berkeley, CA 94720

Dr. Janet Schofield
Learning R&D Center
University of Pittsburgh
Pittsburgh, PA 15260

Dr. Marc Sebrechts
Department of Psychology
Wesleyan University
Middletown, CT 06475

Dr. Judith Segal
Room 819F
NIE
1200 19th Street N.W.
Washington, DC 20208

Dr. Robert J. Seidel
US Army Research Institute
5001 Eisenhower Ave.
Alexandria, VA 22333

Dr. Sylvia A. S. Shafto
National Institute of Education
1200 19th Street
Mail Stop 1806
Washington, DC 20208

Dr. T. B. Sheridan
Dept. of Mechanical Engineering
MIT
Cambridge, MA 02139

Dr. Ted Shortliffe
Computer Science Department
Stanford University
Stanford, CA 94305

Dr. Lee Shulman
Stanford University
1040 Cathcart Way
Stanford, CA 94305

Dr. Randall Shumaker
Naval Research Laboratory
Code 7510
4555 Overlook Avenue, S.W.
Washington, DC 20375-5000

Dr. Miriam Schustack
Code 51
Navy Personnel R & D Center
San Diego, CA 92152

Dr. Robert S. Siegler
Carnegie-Mellon University
Department of Psychology
Schenley Park
Pittsburgh, PA 15213

Dr. Edward Silver
Dept. of Mathematics
San Diego State University
San Diego, CA 92115

Dr. Herbert A. Simon
Department of Psychology
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213

Dr. Zita M Simutis
Instructional Technology
Systems Area
ARI
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Derek Sleeman
Stanford University
School of Education
Stanford, CA 94305

Dr. Edward E. Smith
Bolt Beranek & Newman, Inc.
50 Moulton Street
Cambridge, MA 02138

Dr. Richard E. Snow
Department of Psychology
Stanford University
Stanford, CA 94306

Dr. Elliot Soloway
Yale University
Computer Science Department
P.O. Box 2158
New Haven, CT 06520

Dr. Richard Sorensen
Navy Personnel R&D Center
San Diego, CA 92152

Dr. Kathryn T. Spoehr
Brown University
Department of Psychology
Providence, RI 02912

James J. Staszewski
Research Associate
Carnegie-Mellon University
Department of Psychology
Schenley Park
Pittsburgh, PA 15213

Dr. Robert Sternberg
Department of Psychology
Yale University
Box 11A, Yale Station
New Haven, CT 06520

Dr. Albert Stevens
Bolt Beranek & Newman, Inc.
10 Moulton St.
Cambridge, MA 02238

Dr. Paul J. Sticha
Senior Staff Scientist
Training Research Division
HumRRO
1100 S. Washington
Alexandria, VA 22314

Dr. Thomas Sticht
Navy Personnel R&D Center
San Diego, CA 92152

Dr. John Tangney
AFOSR/NL
Bolling AFB, DC 20332

Dr. Kikumi Tatsuoka
CERL
252 Engineering Research
Laboratory
Urbana, IL 61801

Dr. Perry W. Thorndyke
FMC Corporation
Central Engineering Labs
1185 Coleman Avenue, Box 580
Santa Clara, CA 95052

Dr. Douglas Towne
Behavioral Technology Labs
1845 S. Elena Ave.
Redondo Beach, CA 90277

Dr. James Tweeddale
Technical Director
Navy Personnel R&D Center
San Diego, CA 92152

Dr. Paul Twohig
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Headquarters, U. S. Marine Corps
Code MPI-20
Washington, DC 20380

Dr. Kurt Van Lehn
Department of Psychology
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213

Dr. Beth Warren
Bolt Beranek & Newman, Inc.
50 Moulton Street
Cambridge, MA 02138

Dr. Donald Weitzman
MITRE
1820 Dolley Madison Blvd.
MacLean, VA 22102

Dr. Ronald A. Weitzman
NPS, Code 54Wz
Monterey, CA 92152

Dr. Keith T. Wescourt
FMC Corporation
Central Engineering Labs
1185 Coleman Ave., Box 580
Santa Clara, CA 95052

Dr. Douglas Wetzel
Code 12
Navy Personnel R&D Center
San Diego, CA 92152

Dr. Barbara White
Bolt Beranek & Newman, Inc.
10 Moulton Street
Cambridge, MA 02238

Dr. Christopher Wickens
Department of Psychology
University of Illinois
Champaign, IL 61820

Dr. Mike Williams
IntelliGenetics
124 University Avenue
Palo Alto, CA 94301

A. E. Winterbauer
Research Associate
Electronics Division
Denver Research Institute
University Park
Denver, CO 80208-0454

Dr. Robert A. Wisher
U.S. Army Institute for the
Behavioral and Social Sciences
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Martin F. Wiskoff
Navy Personnel R & D Center
San Diego, CA 92152

Mr. John H. Wolfe
Navy Personnel R&D Center
San Diego, CA 92152

Dr. Wallace Wulfeck, III
Navy Personnel R&D Center
San Diego, CA 92152

Dr. Joe Yasatuke
AFHRL/LRT
Lowry AFB, CO 80230

Dr. Masoud Yazdani
Dept. of Computer Science
University of Exeter
Exeter EX4 4QL
Devon, ENGLAND

Mr. Carl York
System Development Foundation
181 Lytton Avenue
Suite 210
Palo Alto, CA 94301

Dr. Joseph L. Young
Memory & Cognitive
Processes
National Science Foundation
Washington, DC 20550

Dr. Steven Zornetzer
Office of Naval Research
Code 1140
800 N. Quincy St.
Arlington, VA 22217-5000

Dr. Michael J. Zyda
Naval Postgraduate School
Code 52CK
Monterey, CA 93943